Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Matic Volk

Detekcija in sledenje oznake za avtonomno pristajanje brezpilotnega letalnika

magistrsko delo

magistrski program druge stopnje
računalništvo in informatika

Mentor: izr. prof. dr. Danijel Skočaj

Ljubljana, 2017

Zahvala

Zahvaljujem se izr. prof. dr. Danijelu Skočaju za pomoč in vodenje pri izdelavi magistrskega dela. Zahvaljujem se tudi družini in prijateljem, ki so mi stali ob strani in me podpirali na celotni poti študija.

Matic Volk, 2017

Kazalo

Povzetek

Naslov: Detekcija in sledenje oznake za avtonomno pristajanje brezpilotnega letalnika

Brezpilotni letalniki so v zadnjih letih postali pomembno sredstvo za uporabo tako v vojaške kot tudi v civilne namene. V magistrski nalogi razvijemo sistem za detekcijo in sledenje oznake, ki bo pripomogel k realizaciji avtonomnega pristajanja brezpilotnega letalnika. Trenutni letalnik je omejen na pristajanje s padalom, kar onemogoča pristajanje na premikajočih platformah in lokacijah z majhno pristajalno površino. Za odpravo teh težav smo si zastavili cilj razviti sistem, ki bi letalniku omogočili pristajanje v mrežo. Mreža je označena z zasnovano oznako. Razviti sistem smo ovrednotili na posebej pripravljenih videoposnetkih, ki simulirajo realno okolje. Rezultati so pokazali, da je delovanje sistema zadovoljivo pod različnimi pogoji, kot so različni koti, razdalje, osvetljenosti, zamegljenosti, okolja in dodajanje megle v pripravljenih videoposnetkih.

Ključne besede

brezpilotni letalnik, računalniški vid, oznaka, detekcija, sledenje

Abstract

Title: Marker detection and tracking for autonomous landing of unmanned aerial vehicle

In recent years the unmanned aerial vehicles (UAVs) became important assets for military and civil use. In our master thesis we developed a system for marker detection and tracking that enables landing of a UAV without a parachute. The current UAV is limited to landing with a parachute, which makes it difficult to land on moving platforms and small areas. The goal was to develop a system that can help the UAV to land into a net marked with a purposely designed marker. The developed system was evaluated on captured videos that simulate the real environment. Results have shown that the system works well under different circumstances such as various angles, distances, illuminations, blurriness, surrounding areas and fogginess.

Keywords

unmanned aerial vehicle, computer vision, marker, detection, tracking

Poglavje 1

Uvod

1.1 Motivacija

Brezpilotni letalniki so v zadnjih letih postali pomembno sredstvo za uporabo tako v vojaške kot tudi v civilne namene. Brezpilotne letalnike se lahko loči glede na namembnost in vrsto načina krmiljenja. Kvadkopterji so multirotorske naprave, pri katerih je krmiljenje odvisno od vrtenja propelerjev, medtem ko je pri brezpilotnih letalnikih s fiksnimi krili vzlet in krmiljenje nadzorovano z zakrilci ter uravnavanjem hitrosti. Kvadkopterji so bolj primerni za krajše razdalje, lažji za uporabo in omogočajo vertikalno pristajanje ter vzletanje. Letalniki s fiksnimi krili dosegajo višje hitrosti in lahko prepotujejo daljše razdalje. So zelo energetsko učinkoviti, vendar je z njimi težje manevrirati in potrebujejo več prostora za vzlet in pristanek.

V našem delu se bomo osredotočili na brezpilotni letalnik s fiksnimi krili, ki je namenjen za nadzor in izvidnico na morju in bo del opreme ladijskega plovila. Letalnik je prikazan na sliki 1.1. Trenutno je brezpilotni letalnik omejen na pristajanje s padalom. Pristajanje s padalom je neprimerno za pristanek na premikajoči se ploščadi ali lokacijah, ki imajo majhno površino za pristanek. V razvoju je sistem, ki omogoča pristajanje tako, da letalnik prileti v prej postavljeno mrežo. Trenutno ročno krmiljenje letalnika v mrežo je zelo zahteven postopek, potrebnih je veliko veščin ter natančnosti

Slika 1.1: Brezpilotni letalnik s fiksnimi krili [1].

za uspešen pristanek. Rešitev, ki omogoča enostavnejše pristajanje je v razvoju sistema, ki s pomočjo računalniškega vida zmore avtonomno pristati v mrežo. V magistrskem delu se osredotočimo na razvoj algoritmov za detekcijo in sledenje zasnovane oznake, s katero bo mogoče natančno usmerjati letalnik v mrežo za pristajanje.

1.2 Pregled sorodnih del

Problem detekcije in sledenja oznake za namene krmiljenja in pristajanja brezpilotnega letalnika je manj raziskano področje kot pa pri kvadkopterjih. Približek rešitve našega problema je predstavljen v članku [2]. Ker je pristajanje letalnika s pomočjo koordinat GPS nenatančno, so predstavili rešitev pristajanja z vizualnim krmiljenjem. Za potrebe pristajanja so uporabili napihljivo sfero, ki omogoča pristajanje z vseh smeri in ublažitev morebitnih poškodb pri pristanku. Napihljiva sfera je živo rdeče barve, da jo je mogoče enostavno detektirati.

Vizualno krmiljenje je implementirano na osnovi barvne detekcije in sekundarno z metodo momentov. S fotografiranjem rdeče sfere pod različnimi pogoji so določili vizualni spekter, ki določa barvo sfere in na osnovi barvne

klasifikacije implementirali detektor sfere. Za dodatno identifikacijo sfere so uporabili metodo momentov, ki v primeru več rdečih objektov na sliki poskrbi za dodatno klasifikacijo. Z analiziranjem momentov na nekaj primerih so uporabili štiri momente slike za dodatno detekcijo sfere. Prednost pristajanja v napihljivo sfero je ta, da letalnik lahko pristane iz katerekoli strani, če ima maneverski prostor. Pristajanje iz katerekoli strani odpravlja težave zaradi vetra. Če je le mogoče, je dobro, da se pristaja v smeri proti vetru zaradi nižje približevalne hitrosti.

V članku [3] je predstavljena implementacija sistema za avtonomno pristajanje letala z vizualnim krmilnim sistemom. V delu so opisane posamezne faze pristajanja letala. Letalo ima nameščeno kamero na sprednjem delu letala, da lahko posname stezo, ki jo detektor izlušči iz slike. Algoritem za detekcijo pristajalne steze ni opisan, vendar deluje na principu perspektivne projekcije in ohranjanja vzporednih črt, na katere se letalo orientira. Poleg vizualnih podatkov se za krmiljenje uporabljajo še ostali instrumenti, kot sta višinomer in hitrost letala. Testiranje je bilo izvedeno v programu za simuliranje pristajanja letal, kjer so rezultati pokazali, da je mogoče pristati tudi brez pomoči pilota.

Pristop iz dela [4] opisuje implementacijo sistema za pristajanje letalnika s fiksnimi krili, ki krmili letalnik s tal. Kameri za detekcijo in sledenje letalnika sta nameščeni na levi in desni strani pristajalne steze. Z dvema kamerama sistem lahko s pomočjo stereo slike določi 3D koordinate letalnika v prostoru. Prepoznava letalnika deluje na osnovi algoritma AdaBoost, ki s pomočjo šibkih klasifikatorjev zgradi močne klasifikatorje za detekcijo in sledenje letalnika. Za pripravo klasifikatorjev avtorji uporabijo učno množico slik, ki predstavljajo letalnik in negativno učno množico slik. Rezultati so pokazali, da sistem omogoča varno in natančno pristajanje letalnika na pristajalni stezi.

V članku [5] je opisan še en pristop pristajanja brezpilotnega letalnika na pristajalni stezi, pri katerem je kamera nameščena na letalniku. Za izvedbo avtonomnega pristanka igrajo ključno vlogo koordinate GPS in detekcijski

algoritem. Poleg tega uporabljajo tudi informacije o hitrosti, višini in orientaciji letalnika. Glavni del sistema predstavlja model pristajalne steze, iz katerega se izračuna natančna trajektorija pristajanja letalnika. Za postavitev letalnika v model pristajalne steze potrebujejo lokacijo letalnika v realnem okolju. Za natančno lokacijo in priredbo snemane slike v model pristajalne steze poskrbijo podatki o lokaciji GPS, višinomer in detekcijski algoritem steze.

Detekcijski algoritem loči območje pristajalne steze od okolice z barvno metodo v barvnem prostoru HSV. Pridobljen poligon pristajalne steze se transformira v predpripravljen model in tako pridobi transformacijsko matriko med obema slikama. S pomočjo transformaijske matrike nato v realnem času procesirajo predvideno trajektorijo pristajanja letalnika. Sistem so preizkusili na dveh brezpilotnih letalnikih in uspešno opravili pristajanje.

Ker se naša magistrska naloga osredotoča na implementaciji sistema za detekcijo in sledenje oznaki, bomo na kratko predstavili tudi sorodna dela s področja detekcije in sledenja. Večino del, ki uporabljajo detekcijo in/ali sledenje oznaki je implementiranih za manjša okolja in predvsem na krajših razdaljah kot je cilj v naši magistrski nalogi. Članek [6] predstavi različne tehnike detekcije in sledenja objektov z eno kamero. Čeprav se članek osredotoča na obogateno resničnost, so tehnike detekcije in sledenja v osnovi enake kot pri aplikacijah, ki niso namenjene obogateni resničnosti.

Detekcija na osnovi oznake se začne pri zasnovi oznake. Opisana je kvadratna oznaka kot tudi točkovne oznake, ki so produkt stalnega napredka in iskanja rešitev za raznovrstne tipe aplikacij. Točkovne oznake imajo prednost pri detekciji s prekrivanjem, ker jih je mogoče detektirati tudi, če je del točkovne oznake prekrite. Metode detekcije se v osnovi delijo na več korakov. Prvi korak je pridobitev točk interesa v sliki. Poleg točk se uporablja tudi robove za določitev geometrijske oblike objektov v sliki. Obstaja veliko algoritmov za detekcijo točk interesa, kot so Harrisov detektor kotov [7], FAST [8], DoG in Hessian ter mnogi drugi.

Vsako točko interesa je potrebno opisati z opisnikom, kar predstavlja

drugi korak detekcije. Predlagani opisniki so SIFT [9], SURF [10] in CenSurE [11]. Opisniki omogočajo hitro primerjanje ujemanja med predlogo objekta in detektiranim objektom. Kakovost ujemanja se meri z metodami podobnosti in razdalj med dvema opisnikoma. Metode za določanje ujemanj med predlogo in detektiranim objektom so zadnji korak pri detekciji, ki potrdijo ali ovržejo detektiran objekt.

Sledenje objektov v videoposnetku je področje, ki se je v zadnjih letih zelo razširilo. V članku [12] je predstavljen pregled različnih načinov implementacije sledilnikov, ki so danes v uporabi. Avtorji članka razdelijo tipe sledilnih algoritmov na štiri kategorije. Prva kategorija deluje na osnovi ujemanja objektov, ki se glede na opis delijo v podkategorije. Objekte lahko opišemo z regijo, značilkami, deformabilnimi oblikami in vizualnimi ter karakterističnimi modeli objekta.

Naslednja kategorija sledenja deluje na principu filtrov. Eden najbolj znanih filtrov je Kalmanov filter [13], ki deluje tako, da s pomočjo modela stanj napove novo stanje. Vsako stanje se določi glede na napovedano in izmerjeno stanje. Poleg Kalmanovega filtra poznamo tudi filter delcev (ang. particle filter), ki deluje na osnovi metode Monte-Carlo [14]. Modeli stanj se ne držijo linearne ali Gaussove porazdelitve, ampak se prilagajajo glede na uteženost.

Tretja kategorija sledilnikov je opredeljena s klasifikatorji. Problem sledenja rešujejo tako, da z naučenimi klasifikatorji opredelijo, kaj na sliki predstavlja ozadje in kaj iskani objekt. Slabost tega načina je, da je potrebno veliko pozitivnih in negativnih primerov podatkov zato, da se klasifikator nauči razločiti objekt od ozadja.

Zadnja kategorija je fuzijska, ki temelji na kombiniranju različnih sledilnih metod za izboljšanje delovanja sledilnikov. Nekateri primeri fuzijskih sledilnikov temeljijo na uporabi več različnih značilk za opis objekta, drugi se osredotočajo na uporabo več modelov za predstavitev objekta in tretji, ki uporabljajo kombinacijo večih algoritmov tako, da pokrijejo slabosti posameznih algoritmov. Slabost fuzijskih sledilnikov je v tem, da so v večini

primerov časovno zahtevnejši.

Podrobnosti o metodah detekcije in sledenja so natančneje predstavljene v poglavjih 2.2 in 2.3.

1.3 Prispevki

Magistrsko delo zajema razvoj sistema, ki s pomočjo detekcije in sledenja oznaki omogoča krmiljenje brezpilotnega letalnika in s tem popolno avtonomno delovanje. Za potrebe krmiljenja letalnika smo zasnovali oznako, ki nam služi za pozicioniranje in orientacijo letalnika v prostoru. Oznaka je zasnovana tako, da sistem deluje v različnih okoljih in razmerah. Omogočeno je tudi prilagoditi ali zamenjati trenutno zasnovano črno-belo oznako z oznako prilagojeno potrebam posameznih uporabnikov. Oznaka je osnova, na kateri delujeta implementirana algoritma za detekcijo in sledenje. Detekcijski algoritem nam omogoča začetno zaznavo položaja oznake in posredovanje informacij sledilnemu algoritmu, ki nato poskrbi za sledenje oznaki. Sledilni algoritem poskrbi za sledenje oznake na sličicah videoposnetka in vrača pozicijo oznake, ki bo omogočala usmerjanje letalnika. Naloga detekcijskega algoritma je prva detekcija oznake kot tudi ponovna detekcija oznake v primeru, da jo sledilni algoritem izgubi. Cilj razvitega sistema je dodati novo funkcionalnost za pristajanje letalnika, zmanjšati prostor za napake, ki jih lahko povzroči ročno krmiljenje in izboljšati varčevanje z energijo, da omogočimo daljše polete.

1.4 Struktura naloge

Struktura magistrske naloge je poleg uvoda razdeljena na tri glavna poglavja in zaključek. V vsakem poglavju se vsebina razdeli na podpoglavja, ki razčlenijo vsebino in natančneje predstavijo posamezne dele magistrske naloge.

V drugem poglavju je predstavljena teoretična podlaga o računalniškem

vidu, ki je potrebna za razumevanje vsebine v nadaljnjih dveh poglavjih. Teoretična vsebina o računalniškem vidu nam predstavi različne oznake, ki so namenjene za pomoč pri detekciji in sledenju. Drugi del je namenjen osnovam detekcijskih algoritmov z oznako in brez oznake ter tretji del, ki predstavi vrste in osnovne principe delovanja sledilnih algoritmov.

Vsebina tretjega poglavja opisuje zasnovo oznake in implementacijo detekcijskega ter sledilnega algoritma, ki sestavljata celovit sistem. V prvem podpoglavju je opisana zasnova različnih oznak, ki so prilagojene določenim pogojem in okolici, v kateri se nahajajo. Nato sledi opis implementiranega sistema kot celote, ki je v nadaljevanju razčlenjen na podpoglavje o detekcijskem algoritmu in podpoglavju o sledilnem algoritmu. V detekcijskem delu poglavja je podrobno opisana implementacija in delovanje detekcijskega algoritma. Za konec opišemo še implementacijo in delovanje sledilnega algoritma.

Četrto poglavje predstavi eksperimentalno okolje in rezultate implementiranih algoritmov. Ovrednotenje je razdeljeno na več sklopov, ki zajemajo eksperimentiranje na dveh različnih oznakah, ovrednotenje delovanja detektorja in sledilnika posebej ter sistema kot celote. Nadaljnji rezultati se nanašajo le na celotni sistem in predstavljajo simulacijo delovanja sistema pod različnimi pogoji.

V zaključku so opredeljene ugotovitve raziskanega področja, možnosti za izboljšave in nadaljnje delo.

Poglavje 2

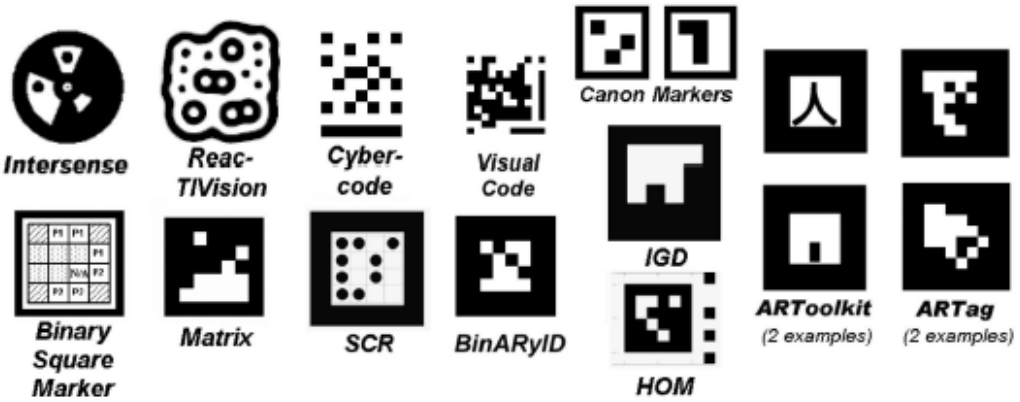
Teoretično ozadje

V tem poglavju je predstavljeno teoretično ozadje računalniškega vida. Za potrebe magistrske naloge se osredotočimo na teoretično podlago o oznakah in detekcijskih ter sledilnih algoritmih, ki delujejo na osnovi zasnovane oznake. Prvo podpoglavje predstavi različne tipe in oblike oznak, njihove slabosti in prednosti. V drugem podpoglavju se osredotočimo na detekcijske algoritme in njihove metode za detekcijo oznak ali detektijo značilk brez uporabe znane oznake. Zadnje podpoglavje predstavi osnovne metode sledilnih algoritmov, prednosti in slabosti ter uporabnost glede na namen in okolico.

2.1 Oznaka

Oznaka je v naprej zasnovana struktura določene oblike, ki znotraj zajete slike predstavlja oznako, na katero se algoritem orientira. Sistem, ki uporablja oznako ali več oznak v kombinaciji z algoritmi za računalniški vid, je namenjen reševanju detekcijskih in identifikacijskih problemov. Take sisteme se uporablja v različnih aplikacijah za računalniški vid, na primer v robotiki, medicini ter za namene obogatene resničnosti. Primeri različnih vrst oznak so prikazani na sliki 2.1.

Pri zasnovi oznake je pomembno, da zagotavlja zanesljivo vizualno referenco v sliki, kar nam omogoča določiti lego oznake in premikanje slike. Oznaka



Slika 2.1: Različne vrste oznak, ki se uporabljajo za različne aplikacije [15].

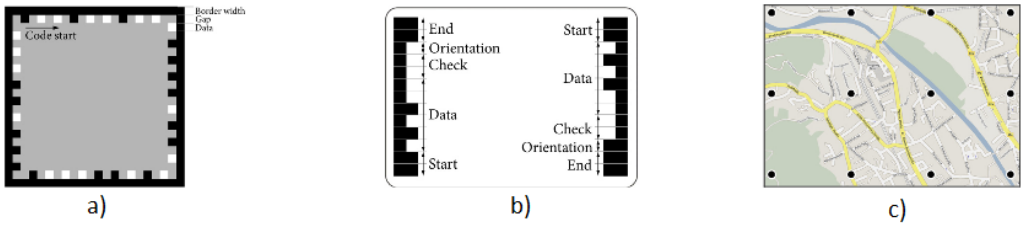
mora biti zasnovana na način, da jo je mogoče z veliko verjetnostjo detektirati tudi pod zahtevnimi pogoji. Kot težke pogoje razumemo [16]:

- prekrivanje,
- razlike v svetlobi,
- veliko oddaljenost,
- odstopanja slike od pravega kota in
- zamegljenosti zaradi hitrega premikanja.

Prekrivanje je problem, ki nastane zaradi ovir na sceni, ki delno ali celotno pokrijejo oznako in tako znatno zmanjšajo uspešnost detekcije. Razen v primerih uniformne svetlobe nastanejo v sliki regije z različno intenziteto slikovnih elementov, ki povzročajo dodatne težave pri predprocesiranju, ko na primer pretvorimo sivinsko sliko v binarno. Razdalja predstavlja problem, če je oznaka premajhna, ter zaradi prevelikih razdalj ni vidna v sliki, da bi jo bilo mogoče detektirati. Različni zorni koti povzročijo deformacijo osnovne oblike oznake, kar lahko povzroči, da iskano obliko oznake v sliki ne najdemo. Zamegljenost slike nastane zaradi nestabilnosti kamere, ki jo povzročijo tresljaji in hitri premiki. V zamegljeni sliki pride do nejasnih linij in nerazločnih

regij. Oznako je potrebno zasnovati glede na potrebe sistema, v katerem bo uporabljen. Sistem naj vsebuje oznako z unikatnim vzorcem tako, da jo algoritem za detekcijo prepozna in locira v procesirani sliki. Vzorci se morajo čim bolj razlikovati od okolice, v kateri se oznaka nahaja [15]. Velikost oznake je osnovna lastnost, ki vpliva na razdaljo, pri kateri je oznako še vedno mogoče zaznati. Pri izbiri barve oznake je potrebno gledati na okolico, v kateri se bo oznaka nahajala. Najbolje je izbrati barvo, ki je čim bolj kontrastna v primerjavi z okoliškimi barvami.

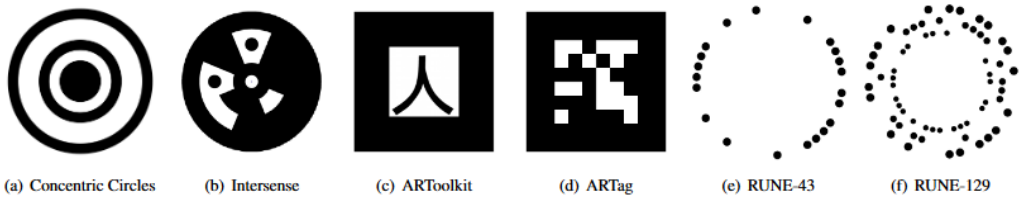
Glede na zasnovo oznake bodo v naslednjih podpoglavjih predstavljene oznake, ki jih ločimo v dve glavni skupini. Prva skupina so oznake s popolnim okvirjem, katera bo zasnovana za potrebe magistrskega dela. Druga skupina oznak so oznake z delnim okvirjem ali celo oznake brez okvirja [17]. Tipi okvirjev so prikazani na sliki 2.2.



Slika 2.2: (a) oznaka z neprekinjenim okvirjem, (b) oznaka z delnim okvirjem in (c) oznaka brez okvirja [17].

2.1.1 Krožna oznaka

Krožna oznaka je oznaka, ki temelji na kontrastnih koncentričnih krožnicah. To pomeni, da je vzorec na ravnini zasnovan z več krožnicami, ki imajo središče v isti točki in si izmenično izmenjujejo dve visoko kontrastni barvi. Klasična oznaka vsebuje črne krožnice na beli podlagi. Okvir krožne oznake predstavlja najbolj zunanja krožnica, ostale krožnice so notranji vzorec oznake. Ta tip oznake spada pod oznake s popolnim okvirjem. Primeri krožnih oznak so prikazani na sliki 2.3 (a), (b), (e) in (f).



Slika 2.3: Oznake, ki se razlikujejo glede na okvir in obliko [18].

Velika prednost krožne oznake je robustnost in enostavnost algoritma za detekcijo le tega. Robustnost se odraža s tem, da se pri detekciji lahko zanašamo na celoten obris, kar omogoča boljšo detekcijo tudi v primerih delnega prekrivanja oznake. Koncentrične krožnice imajo lastnost, da so invariantne na rotacijo in translacijo v vseh treh dimenzijah, kar nam omogoča lažjo detekcijo iz različnih pozicij in rotacij zajete oznake. Pri rotaciji oznake izven planarne ravnine v neplanarno ravnino pride do spremembe krožnic v elipse, ampak hierarhija vgnezdenosti krožnic oziroma elips ostane nespremenjena. S povečevanjem debeline krožnic omogočimo vidljivost oznake na daljših razdalja, ker se pri ožjih krožnicah zaradi oddaljevanja od oznake jasnost linij zmanjša in pride do nerazločnosti med posameznimi krožnicami [19]. Negativne lastnosti krožne oznake se odražajo pri določanju orientacije oznake, saj zaradi invariantnosti ni mogoče določiti rotacije. Rešitev obstaja z uporabo večih oznak ali dodatnih vzorcev znotraj krožnic, ki unikatno določajo orientacijo oznake [20].

2.1.2 Kvadratna oznaka

Kvadratna oznaka je poleg krožne oznake ena od preprostih in zanesljivih rešitev za določanje lokacije in orientacije v sliki. Enako kot pri krožni oznaki je najbolje uporabiti visoko kontrastne barve z razliko, da pri kvadratni oznaki naredimo le en črn kvadrat, ki predstavlja okvir oznake. Kvadratna oznaka je lahko oznaka s popolnim ali delnim okvirjem, ampak za potrebe magistrskega dela se bomo osredotočili le na kvadratno oznako s po-

polnim okvirjem. V notranjosti črnega kvadrata lahko dodamo enega ali več različnih unikatnih vzorcev za enolično določitev oznake. Za določitev orientacije je najbolje, da dodamo v enega od kotov kvadrata dodatno oznako (npr. piko ali kvadratek). Primeri kvadratnih oznak se nahajajo na slikah 2.1, 2.3 (c) in (d). Pri projekciji kvadrata pride do transformacije, pri kateri se iskanje kvadratne oznake v sliki pretvori v iskanje trapeza. Kvadratna oznaka lahko hrani več informacij, katere je tudi lažje izluščiti z razliko od krožne oznake. Poleg tega ima tudi slabosti, ki se nanašajo na lociranje in detekcijo. Pri kvadratni oznaki se osredotočamo na 4 robne točke kvadrata oziroma paralelograma, medtem ko pri krožni oznaki segmentiramo celoten obris krožnice. Razlika je opazna predvsem v primerih, ko pride do prekrivanja oznake in je zato težje določiti robove kvadrata [20].

2.1.3 Oznake z delnim okvirjem in brezokvirne oznake

Poleg oznak s popolnim okvirjem se za nekatere aplikacije uporablja tudi oznake z delnim okvirjem ali celo oznake brez okvirja. Pri oznakah z delnim okvirjem se za potrebe manjše porabe prostora na delovni površini, katero je potrebno detektirati ali slediti, uporabi le črte iz katerih je mogoče določiti rob oznake. Za določitev robov oznake se uporablja enake oziroma podobne metode kot pri oznakah s popolnim okvirjem, saj so črte ponavadi postavljene tako, da predstavljajo obliko kvadrata ali kakšnega drugega preprostega lika. Oznake z nepopolnim robom se uporabljajo za aplikacije, pri katerih vemo, da bo prišlo do prekrivanja oznake. Primer so karte, na katere projeciramo navidezno resničnost in jih zaradi držanja v roki delno prekrivamo s prsti [17].

Omenili smo tudi oznake brez okvirja, ki so namenjeni predvsem detekciji in sledenju večjim površinam. Prednost brezokvirnih oznak je v tem, da prekrivajo zelo majhno površino območja interesa. Ponavadi se uporablja pike ali kvadratke, ki so postavljeni na območje interesa s točno določenim pozicioniranjem. Primer postavitve pik v obliki mreže na zemljevid omogoča dodajanje navidezne resničnosti na območja interesa na zemljevidu [17]. Obe

vrsti oznak sta vidni na slikah 2.1, 2.3 (e) in (f).

2.2 Detekcija

Detekcijski algoritmi za potrebe računalniškega vida so osnova pri zaznavi in prepoznavi iskanega objekta v sliki. Detekcija objektov je proces lociranja enega ali več objektov v sliki. Poleg detekcije objektov v eni sliki lahko detektiramo tudi objekte skozi celotno sekvenco slik videoposnetka. Večina detekcijskih algoritmov uporablja informacije, ki so podane v trenutno procesirani sliki. Nekateri algoritmi uporabljajo poleg informacij pridobljenih v trenutni sliki tudi informacije pridobljene iz predhodnih slik videoposnetka. Način, pri katerem se shranjujejo informacije predhodnih slik, omogoča prilagajanje algoritma in s tem tudi izboljša delovanje.

(a) **(b)**

Slika 2.4: (a) detekcija brez uporabe oznake [21], (b) detekcija na osnovi oznake.

Algoritmi, ki se prilagajajo skozi čas so dinamični, medtem ko algoritmi, ki izvajajo detekcijo s pomočjo informacij, ki jih pridobijo le v trenutni sliki, spadajo med statične algoritme [22].

Detekcijske algoritme v grobem razdelimo na tiste, ki delujejo na osnovi detekcije oznake v sliki, in tiste, ki delujejo brez oznake. Primera detekcije na osnovi oznake in detekcije brez oznake sta prikazana na sliki 2.4. Preden se spustimo v podrobnosti o detekciji na osnovi oznake in detekciji brez uporabe oznake, je potrebno razložiti dva pojma, ki se bosta v nadaljnje večkrat pojavila, in sicer pojem značilka in pojem opisnik značilke.

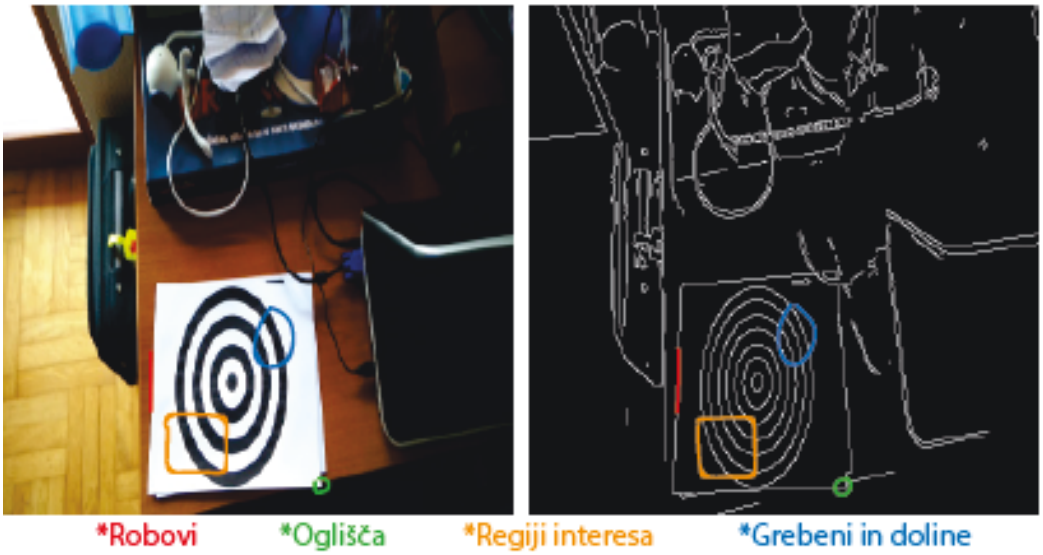
2.2.1 Značilke

Značilke nimajo natančne definicije, ampak so definirane glede na postavljen problem in tip aplikacije, v kateri nastopajo. Značilke v slikah, ki so namenjene za računalniški vid, so točke ali manjše regije v sliki, ki imajo določeno lastnost. So točke interesa, iz katerih izhajamo in jih uporabljamo za namene detekcije in prepoznave iskane regije v sliki. Detekcija značilk je nizkonivojska operacija procesiranja slik. Za detekcijo značilk se je potrebno sprehoditi skozi procesirano sliko in preveriti vsak slikovni element oz. okolico slikovnega elementa ali vsebuje informacije, ki bi lahko predstavljale značilko. V nekaterih primerih je zaradi zahtevnosti algoritma potrebno omejiti iskanje značilk le na določene regije, da prihranimo na času. Večina algoritmov za računalniški vid uporablja detekcijo značilk v sliki kot začeten korak za procesiranje slike.

V ta namen je bilo razvitih veliko različnih algoritmov, ki se razlikujejo predvsem v iskanju določenega tipa značilk, za katere so specializirani. Tipi značilk, ki jih lahko iščemo v sliki so: robovi, koti, regije in grebeni ter doline [23]. Naštete značilke so prikazane na sliki 2.5.

Robovi so točke, ki predstavljajo mejo med dvema ali več regijami v sliki. Obliko roba definira skupina točk v sliki, ki imajo visoko stopnjo gradienta. Robovi se veliko uporabljajo, ker so dovolj preprosti za procesiranje in z njimi pridobimo veliko koristnih informacij. Eden od znanih algoritmov za detekcijo robov v sliki je Canny [24]. Algoritem Canny je podrobneje opisan v poglavju 3.2.1.

Koti kot značilke so točke, pri katerih so zaznane visoke stopnje ukri-



Slika 2.5: Značilke.

vljenosti gradienta v sliki. Prvotno so algoritmi iskali oglišča s pomočjo detektorjev robov, ampak se je s časom izkazalo, da so za to potrebni le gradienti, kar pa je doprineslo do zaznavanja interesnih točk, ki niso tradicionalni koti. Primer zaznane interesne točke, ki ne predstavlja kota, je bela pika na črnem ozadju. Algoritem za iskanje kotov, ki se pogosto uporablja, se imenuje Shi-Tomasi detektor kotov [25].

Regije interesa dopolnjujejo opis strukturam, kot so oglišča, ker se ne osredotočajo le na eno točko, ampak na več točk, ki predstavljajo določeno regijo. Ponavadi regije interesa poleg opisa celotne regije vsebujejo tudi informacijo o lokalnem maksimumu, kar predstavlja center regije interesa. Za razliko od oglišč lahko regije interesa detektirajo območja v sliki, ki so preveč zglajena, da bi jih detektiral algoritem, ki išče oglišča. Za detekcijske agoritme so primerni zaradi razlik v predstavitvi slikovnih struktur. Za iskanje regij interesa se lahko uporablja algoritem MSER (Maximally stable extremal regions) [26].

Poleg naštetih tipov značilk se za v medicinske namene in v slikah iz

zraka uporablja tudi značilke imenovane grebeni in doline. Gre za značilke, ki jih pridobimo s kombiniranjem zgoraj naštetih metod. So podobne regijam interesa, vendar vsebujejo večjo količino informacij ter so namenjene bolj specifičnim namenom. V slikah iz zraka je zato na ta način lažje izluščiti ceste od ostale pokrajine. V medicinskih slikah je ta metoda pridobivanja značilk dobrodošla, saj je na ta način lažje zaznati in predstaviti na primer ožilni ali prebavni sistem.

2.2.2 Opisniki značilk

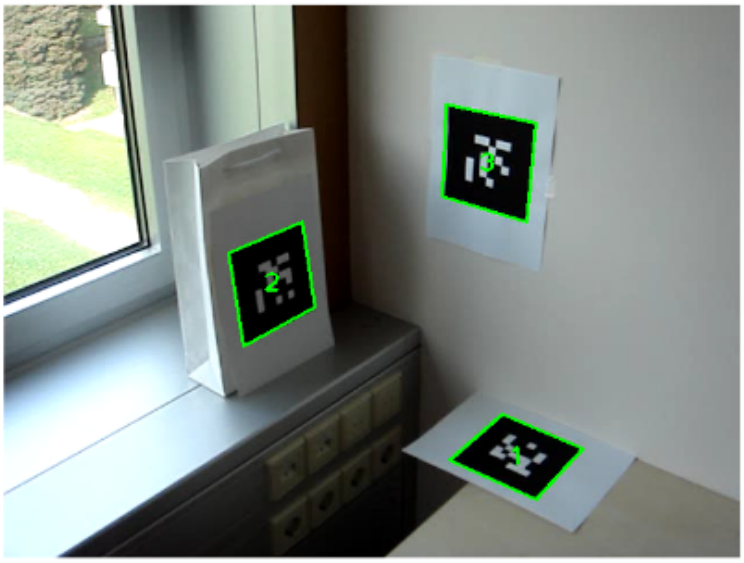
Ko so značilke v sliki detektirane, jih je potrebno primerno shraniti, da se jih lahko uporabi pri primerjanju regije interesa v sliki z referenčno regijo. Pridobljene značilke shranimo tako, da vzamemo lokalno okolico vsake značilke in jo zapakiramo v vektor oziroma deskriptor značilke. Opisnik značilke je struktura, ki nosi lastnosti posamezne značilke. Idealno bi moral biti opisnik značilke unikaten glede ne vse ostale pridobljene opisnike značilk in hkrati enak v vseh pogledih iste točke interesa, iz katere smo pridobili značilko. Vendar je to mogoče le v zelo preprostih primerih, zato vzamemo poleg točke interesa tudi bližnjo okolico in zgradimo opisnik, ki mora čim bolje zadostiti pogoje invariantnosti na osvetljenost, rotacijo in velikost [27].

2.2.3 Detekcija na osnovi oznake

Za potrebe računalniškega vida lahko za namene detekcije uporabimo algoritme, ki detektirajo iskane strukture v sliki, s pomočjo oznake ali večih oznak. Detekcija na osnovi oznake je način, pri katerem je potrebno najprej zasnovati oznako, ki je primerna za uporabo v določenem okolju. Oznake so podrobneje opisane v razdelku 2.1. Glavni nalogi oznake, ki jo vstavimo v sceno, sta določanje relativne lokacije in orientacije struktur v sliki. Glede na sistem, v katerem bo oznaka uporabljena, je nato potrebno implementirati algoritem, ki bo zaznal oznako in iz nje tudi izluščiti želene informacije. Detekcijski algoritmi na osnovi oznak se veliko uporabljajo v industriji, robotiki

in navidezni resničnosti. Oznake se uporablja, ker je tako najlažje izluščiti želene informacije, saj so prilagojene za enostavno prepoznavo v okolici, za katero so zasnovane.

Prednosti detekcije na osnovi oznake se izkažejo v okoljih, kjer vsebujejo slike velike uniformne regije, dinamične teksture in odsevne površine. Pridobivanje pravilne velikosti iskane strukture je enostavnejše, saj imamo podatke o velikosti oznak. Oznake lahko vsebujejo dodatne informacije, kot so identifikacija, besedilo itd. Detekcijski algoritmi na osnovi oznak so ponavadi računsko manj potratni in porabijo manj procesorske moči ter pomnilnika. Primer rezultata detekcijskega algoritma za detekcijo oznake je prikazan na sliki 2.6.



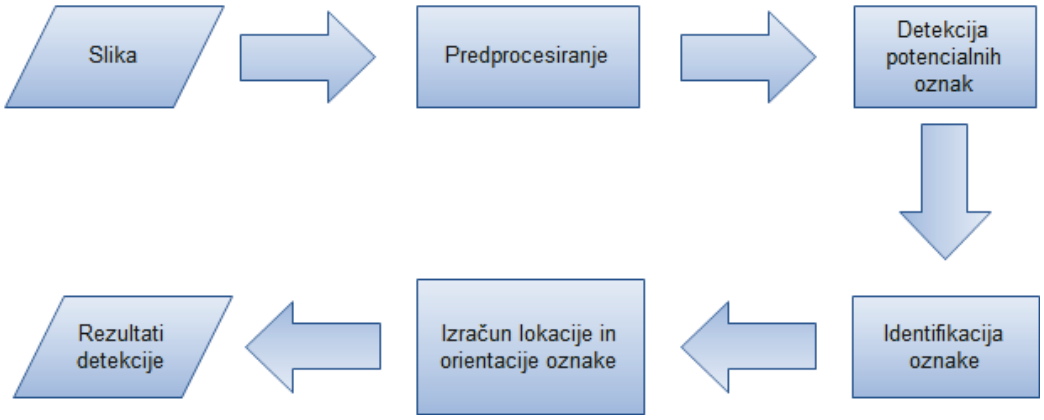
Slika 2.6: Detekcija večih oznak [28].

Algoritmi za detekcijo oznak so prilagojeni specifični oznaki, se pa večinoma vsi držijo enakega osnovnega postopka. Postopek detekcije oznake lahko razdelimo na štiri osnovne korake:

- predprocesiranje,
- detekcija potencialnih oznak,

- identifikacija oznake in
- izračun lokacije in orientacije oznake.

Osnovni koraki detekcije oznake so prikazani na sliki 2.7. Predhodni korak vključuje zajem slike ali videoposnetka, na katerem izvajamo detekcijo.



Slika 2.7: Potek detekcije oznake.

Predprocesiranje je ključnega pomena za predpripravo slike, da je potem proces detekcije enostavnejši. V koraku predprocesiranja procesirano sliko ponavadi pretvorimo v sivinsko, poskušamo čim bolj izločiti šum in nato uporabimo eno od metod za pridobivanje značilk ali uporabimo preprosto metodo za binariziranje slike z določenimi pragom. Metoda binarizacije lahko uporablja prilagodljiv prag, ki se bolje obnese pri slikah z večimi spremembami v osvetljenosti.

Iz predprocesirane slike se nato izloči regije, ki očitno niso oznaka. To so recimo premajhne regije ali regije, ki so popolnoma različne od oblike oznake. Preostale regije se označi kot potencialne oznake. Za pridobitev prave oznake je potrebno iti skozi postopek izločevanja. Glede na lastnosti oznake lahko predvidevamo in preverimo ali so v sliki regije z enakimi lastnostmi. Za kvadratne oznake se išče kvadrate in štiri oglišča, za okroglo oznako se poskuša pridobiti regije, ki so v obliki elips. Poleg tega je pomembna velikost regije in notranjost oznake.

Identifikacija oznake je odvisna tudi od vzorca, ki ga nosi oznaka znotraj njegovih robov. Znani algoritmi za iskanje določenih oblik oznak so Harrisov detektor za detekcijo kotov, Houghova transformacija za iskanje linij v sliki in krožni za iskanje krogov v sliki. Ko je detekcija in identifikacija oznake zaključena, nastopi še zadnji korak, ki določi pozicijo in orientacijo oznake. Lokacijo določimo s tremi koordinatami x - širina, y - višina, z - globina. Orientacija je predstavljena s tremi koti α - rotacija okrog osi y, β - rotacija okrog osi x in γ - rotacija okrog osi z. Pozicijo kamere oziroma oznake je mogoče določiti z minimalno štirimi nekolinearnimi točkami, ki ležijo na isti ravnini. Pri kvadratni oznaki se lahko osredotočimo le na štiri oglišča in iz njih pridobimo relativno pozicijo oznake glede na kamero. Pri določanju pozicije je potrebno paziti na transformacijo koordinat kamere v koordinate sveta in obratno.

Slika zajeta s kamero ima karakteristike, kot so goriščna razdalja, orientacija kamere in velikost oznake v sliki. Te karakteristike določajo transformacijo med koordinatami. Ko pridobimo informacijo o relativni poziciji oznake, je proces detekcije zaključen. Uporaba pridobljenih informacij se razlikuje glede na tip aplikacije. Nekatere aplikacije poleg pozicije oznake potrebujejo še informacije, ki so shranjene v vzorcu oznake in se pridobijo po enakem principu kot za detekcijo same oznake [29].

2.2.4 Detekcija brez uporabe oznak

Poleg detekcije na osnovi oznak lahko uporabimo tudi alternativne metode detekcije, ki se ne zanašajo na oznako, ampak uporabljajo informacije, ki so na voljo v sliki brez dodajanja umetnih referenčnih informacij. Vsaka slika ima različne lastnosti, ki jih lahko opišemo s pomočjo značilk. Značilke so podrobneje opisane v razdelku 2.2.1. Način detekcije brez uporabe oznak je namenjen aplikacijam, ki jim okolje ne omogoča uporabo oznak za detekcijo. Poleg tega so oznake zelo občutljive na prekrivanje, zato je možna rešitev uporabiti informacije, ki se že same po sebi nahajajo v sliki. Za uspešno izvajanje detekcije brez uporabe oznake še vedno potrebujemo nekakšen opis

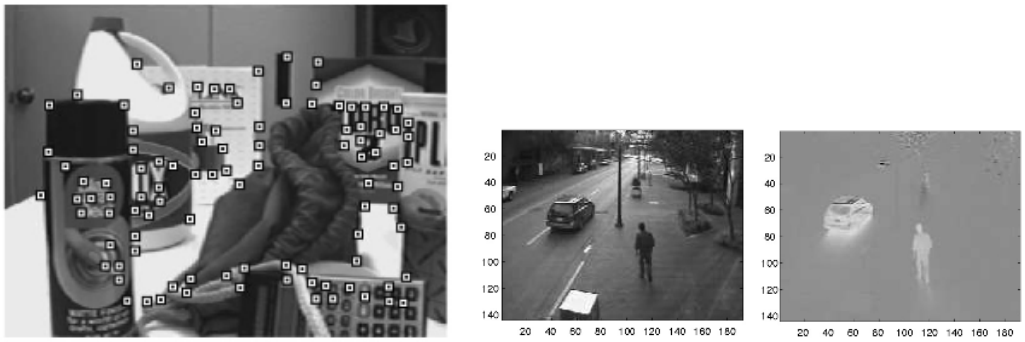
oziroma model, ki predstavlja iskano strukturo v slikah, da ga lahko prepoznamo in izluščimo iz njih.

Prednost modela iskane strukture je v tem, da ni potrebno vstavljati umetne izdelane oznake v sceno, da bi zaznali iskano strukturo, ampak lahko zaznamo strukturo le s pomočjo njenega modela, ki predstavlja lastnosti, ki jih je potrebno poiskati v sliki in na ta način prepoznati iskano strukturo. Slabost iskanja struktur v sliki je zahtevnejše procesiranje, kar porabi več časa za detekcijo, zato je vedno potrebno oceniti primernost algoritma, ki ga bomo uporabili za namene reševanja določenega problema. Detekcijske algoritme brez uporabe oznake lahko razdelimo glede na način detekcije v več kategorij [22]:

- točkovna detekcija,
- detekcija z odstranjevanjem ozadja,
- detekcija s segmentacijo in
- detekcija s nadzorovanim učenjem.

Točkovni detektorji se uporabljajo za iskanje značilk, ki imajo določeno teksturo na določeni točki v sliki. Značilnost značilk je, da so matematično dobro definirane, imajo natančno določeno pozicijo v sliki in vsebujejo enolične informacije, so robustne glede na lokalne in globalne deformacije v sliki in dovolj raznoliko izbrane, da jih ni mogoče zamešati med seboj. Na sliki 2.8 (a) je prikazana detekcija točkovnih značilk. Znani točkovni detektorji v literaturi so detektor KLT [30], detektor SIFT [9], Harrisov detektor točk [7] in nekateri drugi [31].

Pri detekciji objektov z odstranjevanjem ozadja nam že samo ime pove, da gre za način, pri katerem je glavni del detekcije prepoznati in ločiti ozadje od iskanega objekta. V literaturi najdemo veliko različnih načinov odstranjevanja ozadja, ampak se večinomo vsi držijo štirih glavnih korakov. Prvi korak je preprocesiranje vhodne slike tako, da odstranimo šum in pripravimo sliko za nadaljnje procesiranje. Drugi korak poskrbi za pripravo modela ozadja,



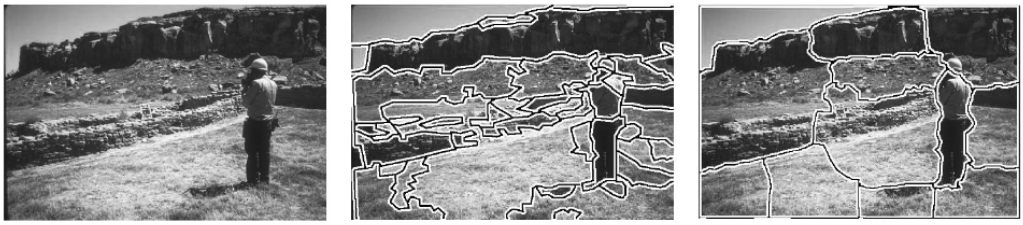
(a) **(b)**

Slika 2.8: (a) točkovna detekcija, (b) detekcija z odstranjevanjem ozadja [31].

ki predstavlja okolico iskanega objekta. V tretjem koraku identificiramo slikovne elemente, ki ne sovpadajo z modelom ozadja in jih nato v četrtem koraku potrdimo oziroma označimo kot ospredje z določeno mejo zanesljivosti. Izhod je maska, ki v sliki ohrani le iskani objekt [32]. Primer detekcije z odstranjevanjem ozadja je prikazan na sliki 2.8 (b).

Cilj detekcije s segmentacijo je razdeliti sliko na regije tako, da si piksli posamezne regije delijo čim bolj podobne lastnosti in nato identificirati iskano strukturo. Vsak segmentacijski algoritem mora rešiti dva ključna problema. Prvi problem je postaviti primeren kriterij za določitev posamezne regije. Drugi problem opredeljuje učinkovito rešitev za razdeljevanje slike na regije. Nekatere od metod za segmentacijo slik so gručenje s povprečnim premikom (Mean-Shift Clustering) [33], z rezanjem grafov (Graph-Cut) [34] in z metodo aktivnih obrisov (Active contours) [35]. Prva metoda išče gruče v predpripravljenem prostoru, ki vsebuje informaciji o lokaciji in barvi posameznega slikovnega elementa. Rezanje grafov deluje na principu razdelitve slike na manjše regije. Regije, ki niso del iskane celotne regije se odstrani glede na lastnosti, kot so barva tekstura in osvetljenost. Pri metodi aktivnih obrisov se zaključen obris ovije okrog roba strukture tako, da zaključi njeno regijo. Ovijanje deluje s pomočjo energijske funkcije, ki se prilagaja glede na hipo-

tetično regijo strukture [31]. Primer detekcije s segmentacijo je prikazana na sliki 2.9.



(a) (b) (c)

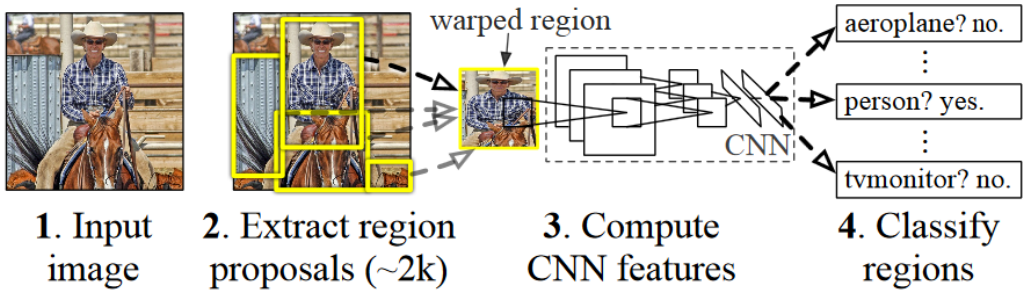
Slika 2.9: Potek detekcije s segmentacijo slike [31].

Nadzorovano učenje je metoda detekcije, pri kateri se za namene iskanega objekta shrani različne variacije pogledov objekta in nauči algoritem, da iz vseh najdenih objektov oziroma oblik s pomočjo klasifikacije izlušči iskani objekt. Pomembno vlogo pri detekciji objektov in učenju klasifikatorja igrata razred in značilke objekta, ki se določita vnaprej. Razred objekta je predstavljen kot tip objekta, ki ga iščemo v sliki. Razredi so globalne predstave, kot so avtomobili, drevesa, sadje ... Značilke so namenjene, da opredelijo lastnosti in detajle, ki enolično določajo iskani objekt. Pristopi, ki delujejo na principu učenja, vključujejo nevronske mreže, odločitvena drevesa ali strojno učenje, vendar ne nujno v vseh primerih. Nadzorovane metode učenja ponavadi potrebujejo veliko količino vzorcev, da lahko izvajajo učenje z označevanjem vzorcev v različne razrede in na koncu identificirajo iskani objekt s klasifikacijo [31].

2.2.5 Novejše metode detekcije

V zadnjih letih se je zaradi zmogljivejše računalniške opreme razvila veja detekcijskih algoritmov, ki temeljijo na globokem učenju. Pri globokem učenju je vir uspešnosti detekcije velika količina podatkov. Detekcijo opravljajo nevronske mreže, ki se najprej naučijo detektirati enega ali več razredov objektov s pomočjo učne množice [36]. Za računalniški vid, pri katerem so

večinoma podatki predstavljeni s slikami, se pogosto uporablja konvolucijske nevronske mreže (ang. Convolutional neural network - CNN). Glavno vlogo delovanja katerekoli nevronske mreže določa njena arhitektura. Konvolucijska nevrnoska mreža je sestavljena iz vhodnega in izhodnega sloja ter dodatnih skritih slojev. Skriti sloji opravljajo naloge, ki zajemajo konvolucijo, združevanje (pooling), uteževanje in normalizacijo [37].



Slika 2.10: Postopek detekcije s pomočjo konvolucijske nevronske mreže [38].

Detekcija objetkov s konvolucijsko nevronsko mrežo deluje tako, da je najprej potrebno nevronsko mrežo naučiti razliko med iskanim objektom in ostalimi objetki ter okolico. Kot je že bilo omenjeno, je za to potrebna velika količina podatkov, ki se jih nato loči na učno in testno množico. Obe množici so anotirane tako, da vsebujejo pozitivne in negativne primere iskanega objekta. Nevronska mreža se na učni množici uči detektirati iskane objekte na sliki, medtem ko se testno množico uporabi za preverjanje učinkovitosti nevronske mreže pri detekciji objektov.

V članku [38] je predstavljena konvolucijska nevronska mreža, ki izvaja detekcijo objetkov na osnovi segmentacije. Segmentacija poteka tako, da sliko razdeli na veliko število manjših regij, iz katerih se izlušči značilke. Detekcija se zaključi s klasificiranjem regij, ki so označene kot pozitivne, če predstavljajo iskani objekt ali negativne, če regija ne predstavlja iskanega objekta. Postopek delovanja detekcije je prikazan na sliki 2.10. Za izboljšanje delovanja nevronske mreže so avtorji s pomočjo nadzorovanega učenja nastavili

parametre posameznih slojev v arhitekturi nevronske mreže. Ta način uporabe konvolucijskih nevronskih mrež je zelo razširjen in se lahko uporablja za detekcijo različnih objektov, pri čemer je potrebno arhitekturo oz. nastavitve slojev prilagajati specifičnim potrebam.

2.3 Sledenje

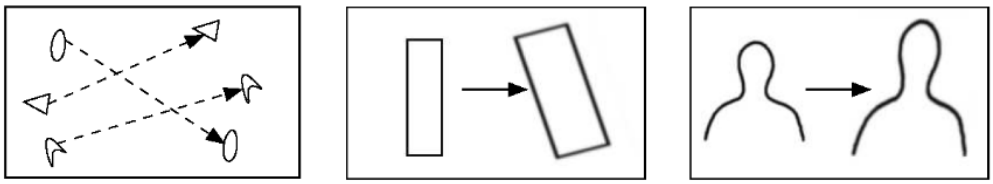
Sledenje predstavlja določitev lokacije, poti in karakteristik točke ali regije interesa s pomočjo meritev pridobljenih z enim ali več senzorji. V računalniškem vidu se algoritmi za sledenje večinoma osredotočajo na sledenje regij interesa v videoposnetkih [39]. Cilj algoritma za sledenje je generirati trajektorijo, ki predstavlja potovanje regije interesa skozi sekvenco sličic videoposnetka. Na sledenje lahko gledamo z dveh vidikov. Prvi vidik predstavlja sledenje in lociranje enega ali več objektov na fiksni sceni, medtem ko drugi vidik predstavlja sledenje na premikajoči sceni s premikajočo regijo interesa [40]. Da se lahko sledenje začne izvajati, je potrebno določiti začetno stanje regije interesa, za katero poskrbi detektor ali ročna označitev, ki jo potem podamo kot vhodni parameter v algoritem za sledenje.

Algoritme za sledenje lahko razdelimo na več načinov. Lahko jih delimo na sledilnike, ki se izvajajo v času zajema videoposnetka in sledilnike, ki se izvajajo v nadaljnji obdelavi že zajetega videoposnetka. Sledilniki, ki se izvajajo v času zajema videoposnetka, se uporabljajo za takojšnje sledenje pri nadzornih kamerah, ki snemajo promet, ljudi, živali ali celo neko proizvodnjo v tovarni. Sledenje v kasnejši obdelavi videoposnetka je namenjeno za analizo tekem, spremembam v naravi in ostalim analitičnim raziskavam. Pri naknadnem sledenju je več manevrskega prostora, ker imamo več časa in možnost sprehajanja po posnetku naprej in nazaj ter po potrebi sprotno spreminjati parametre sledilnega algoritma, vendar je včasih potrebna informacijo zajeti v času zajemanja slike ali videoposnetka in za te namene se uporablja takojšnje sledenje.

Naslednja delitev sledilnih algoritmov se nanaša na dolgoročne in krat-

koročne sledilnike. Glavna razlika med kratkoročnimi in dolgoročnimi sledilniki je v tem, da ko kratkoročni sledilnik izgubi sled iskane regije interesa, nima možnosti ponovne detekcije in vrnitve na pravo sled. Dolgoročni sledilniki imajo običajno poleg sledilnika še sistem, ki omogoča, da se sledilnik ponastavi, če izgubi sled regije interesa. To odpravi tako, da ima implementiran tudi detektor, ki ponovno inicializira stanje sledilnika in tako omogoči nadaljnje sledenje.

Algoritme za sledenje lahko razdelimo še na generativne in diskriminativne metode. Generativne metode uporabljajo modele izgleda regije interesa, ki opisujejo lastnosti regije in hranijo potrebne informacije za pravilno delovanje sledilnika. Diskriminativne metode imajo drugačen pristop, in sicer delujejo na principu iskanja meje med regijo interesa in okolico. Diskriminativni sledilniki se osredotočajo na ločevanje ozadja od iskanega objekta in mu na ta način sledijo skozi sekvenco sličic videoposnetka [41].



Slika 2.11: (a) Točkovni vizualni model, (b) jedrni vizualni model, (c) vizualni model obrisa [31].

Pri sledenju je zelo pomembno, kako predstaviti regijo interesa. Za predstavitev regije interesa se zato uporablja vizualni model, ki hrani informacije o vizualni podobi regije, ki ji želimo slediti. Sledenje lahko razdelimo glede na vizualni model v tri kategorije:

- točkovno sledenje,
- sledenje jedra in
- sledenje obrisa.

Vsako od sledenj se potem deli še naprej v podkategorije, ki so specifične za določene primere sledenja in predstavitve regije interesa. Na sliki 2.11 je pod (a) prikazan točkovni model, (b) prikazuje jedrni model in (c) model obrisa. Vsi trije vizualni modeli in ustrezni sledilni algoritmi bodo predstavljeni v naslednjih podpoglavjih.

2.3.1 Točkovno sledenje

Sledenje lahko predstavimo kot detekcije regij interesa, predstavljene s točkami v vsaki od sličic videoposnetka. Metode, ki temeljijo na točkovni predstavitvi regije interesa, se delijo na deterministične in statistične. Obe metodi upoštevata naslednje omejitve glede ujemanja točk skozi posamezne sličice videoposnetka:

- skozi sekvenco sličic se lokacija regije interesa drastično ne spreminja,
- spremembe v smeri in hitrosti premikanja regije interesa so majhne in
- regija interesa se ne deformira skozi čas.

Deterministične metode za iskanje ujemanja točk med sosednjima sličicama uporabljajo kvalitativno gibalno hevristiko. Določanje ujemanja v trenutni sličici **t** se meri s ceno ujemanja glede na prejšnjo sličico **t-1**. Izbira najboljše točke v sličici se med vsemi možnimi točkami izbere optimalno rešitev, ki se določi glede na zgoraj naštete omejitve [31]. Primer algoritma, ki deluje na principu deterministične metode in upošteva omejitvi za majhno spremembo lokacije in nedeformiranje regije vzame zaporedni sličici in uporabi kriterij najbližjih sosedov. Za določitev regije interesa algoritem uporabi več točk, iz katerih za vsako posebej generira trajektorijo [42].

Točka interesa v videoposnetku je podvržena naključnim spremembam stanja zaradi šuma v videoposnetku. Spreminjata se hitrost premikanja in oblika. Oba parametra stanja točke interesa se upoštevata pri statističnih metodah ujemanja. Glavna lastnost iskanja ujemanja točk med sosednjima

sličicama pri statističnih metodah je, da se upošteva model nedoločenosti, ki je predstavljen z normalno porazdeljenim šumom [31].

Točkovni sledilniki so primerni za sledenje majhnim objektom, ki jih lahko predstavimo z eno točko. V primeru sledenja večjih objektov se lahko uporabi več točk, ki so porazdeljene čez objekt in se držijo določenih pravil za enoličnost točk. Ponavadi se predpostavlja, da so točke postavljene na togo telo, kar poenostavi segmentacijski proces [31]. Na sliki 2.12 je prikazano sledenje večih točk, ki je predstavljeno s pomočjo trajektorij posamezne točke.

Slika 2.12: Rezultat točkovnega sledenja, ki je predstavljen s trajektorijami posameznih točk [31].

2.3.2 Sledenje jedra

Algoritmi za sledenje jedra se osredotočajo na procesiranje premikanja regije interesa med sosednjimi sličicami videoposnetka. Regija interesa oziroma iskani objekt je predstavljen s primitivnimi geometrijskimi oblikami. Algo-

ritmi, ki sledijo jedrom, se razlikujejo po predstavitvi izgleda objekta, številu sledenih objektov in metodah za določitev premikanja objektov. Metode za sledenje jedra so razdeljene na dve podkategoriji. Prva kategorija se osredotoča na različne modele predstavitve regije interesa, medtem ko druga podkategorija uporablja modele izgleda z različnih pogledov. Slika 2.13 prikazuje jedrno sledenje skodelice, ki je predstavljena z elipso.

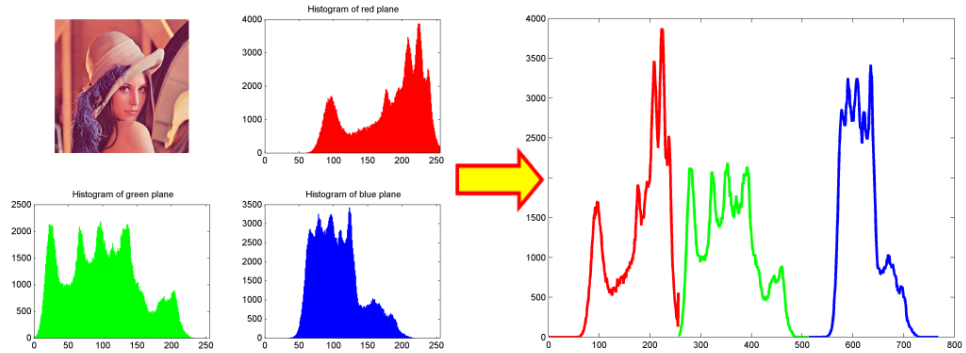
Slika 2.13: Sledenje skodelice s pomočjo sledenja jedra [43].

Regijo interesa lahko predstavimo na različne načine. Najbolj pogosta uporaba predstavitve modela izgleda so predloge regije interesa (ang. template) in modeli na osnovi gostote pojavitve določene lastnosti (ang. density-based). Modela sta enostavna za uporabo in računsko preprosta, ker porabita malo procesorske moči. Metoda ujemanja s predlogo regije interesa je preprosta metoda, ki preveri vse lokacije na sliki in s pomočjo mere podobnosti ali razdalje določi podobnost ali različnost med predlogo in trenutno primerjano regijo. Nekaj od najbolj uporabljenih mer podobnosti in razdalj [44]:

- Evklidska razdalja
- križna korelacija
- cosinususna podobnost
- Hammingova razdalja
- Bhattacharyya razdalja
- Hellingerjeva razdalja

Obstaja še mnogo drugih mer podobnosti in razdalj, ki so ravno tako pomembne, ampak se uporabljajo manj pogosto. Poleg predlog se uporablja tudi druge modele, ki predstavijo regijo interesa z značilkami. Uporabljajo se

sivinski in barvni histogrami, gradientne slike regije interesa ali celo mešanica večih različnih značilk. Primer predstavitve predloge s pomočjo barvnega histograma je prikazan na sliki 2.14. Poleg neposrednih podatkov, ki jih lahko vsebuje slika ali regija interesa, se lahko zgradi kompleksnejše modele izgleda, ki uporabijo dane podatke in iz njih izračunajo povprečja, deviacije, mediano in ostale statistične meritve. Optični tok je še ena možna rešitev, ki deluje na osnovi smeri gibanja objekta in ravno tako omogoča sledenje skozi sličice videoposnetka [31].



Slika 2.14: Vizualni model opisan s barvnim histogramom [45].

Sledenje s pomočjo predloge objekta ima pomanjkljivost, ker se lahko pogled iskanega objekta skozi čas spreminja in zaradi tega povzroči napako pri sledenju, ker model izgleda ni veljaven oziroma ne predstavlja več iskanega objekta. Za rešitev problema prve podkategorije se uporablja aktivne modele izgleda, ki uporabljajo več pogledov objekta [31]. Modeli izgleda se generirajo s pomočjo rekonstrukcije ali dejanskega zajema iz različnih zornih kotov, še preden začnemo s sledenjem želenega objekta. Vsak model izgleda zajema določen zorni kot, ki je naučen z označenimi slikami različnih orientacij in pozicij. Algoritem izbere najprimernejšega izmed zbirke modelov izgleda objekta in s pomočjo parametrov, ki jih hrani izbrani model, lažje sledi in se prilagaja novim situacijam skozi sekvenco sličic videoposnetka. Poleg pozitivnih modelov lahko algoritem zajame tudi negativne, ki pred-

stavljajo okolico regije interesa in s tem lažje filtrira pravilne zadetke od napačnih [46].

Glavni cilj algoritmov za sledenje jedra je določitev gibanja objekta. Objekti so predstavljeni z regijo, ki jo ponavadi določajo enostavne geometrijske oblike in če določimo njihovo gibanje skozi sličice videoposnetka lahko določimo tudi njihovo regijo in lastnosti. Obe lastnosti gibanje in predstavljenost objekta sta povezana med seboj ter oba prispevata k boljšemu sledenju objekta. Ena od omejitev osnovnih geometrijskih oblik za zajem objekta je, da lahko ostanejo deli objekta izven geometrijske oblike ali pa je zajeta tudi okolica okrog objekta. Za odpravo tega problema obstaja več rešitev. Ena od rešitev je, da je zajeta regija vedno znotraj objekta, druga možna rešitev uporablja uteži, ki so porazdeljene glede na verjetnostno funkcijo barv in teksture posameznih slikovnih elementov [31]. V naslednjem poglavju bo predstavljena še ena rešitev, ki odpravi težave pri zahtevnejših oblikah objekta in se imenuje sledenje obrisa objektov.

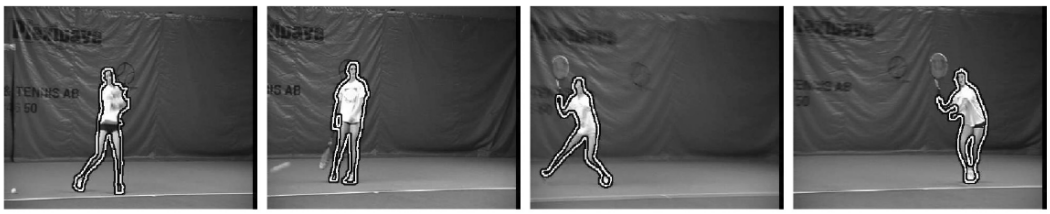
2.3.3 Sledenje obrisa

Sledenje obrisa je namenjeno regijam interesa oziroma objektom, ki so kompleksnejših oblik, kot so na primer dlani ali celotno telo in jih ni mogoče učinkovito predstaviti s preprostimi geometričnimi oblikami. Obrisi natančno opišejo obliko objekta, ki ga predstavlja zgrajeni model s pomočjo barvnega histograma, robov ali obrisa objekta. Algoritme za sledenje obrisov delimo v dve kategoriji. Prva kategorija deluje na principu primerjanja ujemanja oblike objekta, medtem ko se druga kategorija osredotoča na sledenje obrisov v sekvenci sličic videoposnetka [31].

Princip primerjanja oblik objekta je podoben kot pri primerjavi predloge regije s tem, da je tukaj regija predstavljena s polnim likom poljubne oblike. Ravno tako je potrebno zgraditi model, ki predstavlja in hrani lastnosti iskanega objekta. Model se skozi sekvenco sličic prilagaja spremembam izgleda. Po lokalizaciji iskane oblike objekta se model posodobi glede na novo najdeno obliko, kar pripomore k boljšemu sledenju zaradi sprememb zornega

kota, svetlobe in prožnosti objekta. Primerjava oblik je lahko tudi na način, ki je opisan v poglavju točkovnega sledenja 2.3.1 le, da se tukaj uporablja celotno regijo objekta in ne samo posamezne točke. Za predstavitev oblike objekta se večinoma uporablja histograme barv ali robov. Ujemanje med dvema zaporednima sličicama se preverja med predlogo modela in trenutno pridobljenim modelom. Nivo ujemanja med modeloma se izračuna s pomočjo podobnostnih funkcij ali funkcij razdalje [31].

Druga kategorija sledenja se osredotoča na sledenju obrisa brez regije znotraj obrisa objekta. Primer sledenja obrisa telesa je predstavljen na slikah 2.15.



Slika 2.15: Sledenje obrisa telesa [31].

Pri sledenju obrisov se zgradi začetno stanje obrisa, s katero se potem izvaja sledenje na način razvijanja obrisa iz prejšnje sličice na naslednjo tako, da iterativno prilagaja obris, dokler se ne prileže novemu obrisu objekta. Za delovanje tega algoritma je potrebno, da se vsaj del obrisa prekriva z objektom, ki ga iščemo v trenutni sličici. Za sledenje z razvijanjem obrisa objekta se uporabljata dve metodi. Prva metoda temelji na posodabljanju modela, ki hrani trenutno stanje oblike objekta in upošteva lokacijo ter gibanje objekta. Stanje se posodablja vsakič, ko se določi maksimalna verjetnost ujemanja oblike objekta s predlogo oblike objekta. Druga metoda sledenja z razvijanjem obrisa deluje na principu minimizacije energije stanja. Metodi za minimizacijo energije sta lahko požrešna metoda ali metoda padajočega gradienta. Energija obrisa je definirana kot trenutna informacija ali trenutni gradient, ki je ponavadi zgrajen s pomočjo optičnega toka [31].

Sledenje oblike objekta se uporablja predvsem v primerih, ko so pomembne informacije v celotni regiji sledenega objekta. Nekateri algoritmi uporabljajo le obris objekta za sledenje, nekateri pa celotno regijo. V glavnem so algoritmi, ki uporabljajo celotno regijo za sledenje natančnejši in robustnejši na šum. Prednost sledenja obrisov se pokaže v fleksibilnosti prilagajanja različnih oblik objektov. Za predstavitev objektov, ki so sledeni na principu obrisov se lahko uporablja modele [31]:

- gibanja,
- izgleda,
- oblike in
- kombinacije le teh.

Odpravljanje težav pri sledenju pod vplivom prekrivanja sledenega objekta se algoritem najbolj zanaša na lastnosti konstantnega premikanja in pospeševanja, kar omogoča določitev nove hipotetične lokacije objekta. Kot pri ostalih algoritmih, ima sledenje obrisov tako prednosti kot slabosti, ampak vsak algoritem je v nekaterih dober, v drugih pa ne, zato je potrebno izbrati algoritem za sledenje, ki je najbolj primeren za določen problem [31].

2.3.4 Novejše metode sledenja

Novejše metode sledeneja so tako kot pri detektorjih osredotočajo na konvolucijske nevronske mreže. Delovanje sledilnikov, ki delujejo na osnovi konvolucijskih nevronskih mrež, je v veliki meri odvisno od učenja nevronske mreže na učni množici podatkov. Druge razlike se pojavljajo pri sami sestavi slojev nevronske mreže in njihovih nastavitvah. Ena od novejših konvolucijskih nevronskih mrež, ki je predstavljena v članku [47], se uči na učni množici velikega števila različnih domen. Domene se sekvence videoposnetkov, ki ponazarjajo različne predstavitve iskanega objekta. Avtorji predstavijo način, pri katerem uporabijo eno, prej omenjeno nevronsko mrežo, za učenje skupnih slojev, ki jih uporablja še druga nevronska mreža. Prej naučeni sloji nevronske mreže se nato uporabijo v drugi nevronski mreži, ki pa poleg teh slojev vsebuje še dodatne sloje, ki se učijo klasifikacije objekta kar v času

izvajanja sledilnika. Klasifikacija se izvaja nad okni, ki so postavljena okoli zadnje znane pozicije iskanega objekta na sliki.

Poleg nevronskih mrež se še vedno uporabljajo klasične metode opisane v prejšnjih poglavjih, vendar so nadgrajene z novimi pristopi in metodami, ki znatno izboljšajo delovane sledilnikov. Veliko sledilnikov se poslužuje uporabo večih modelov za predstavitev objekta. Večinoma se uporabljata modela, kot sta gibalni model in model izgleda [48]. Drugi sledilniki nadomestijo predstavitev objetka z eno predlogo (ang. patch) tako, da razdelijo predlogo na več manjših. Vsaka od predlog je potem opisana z različnimi opisniki, kot so HOG, SIFT [9], SURF [10] in ostali. Eden od algoritmov za sledenje, ki uporablja več predlog za predstavitev objekta, je predstavljen v članku [49].

Poglavje 3

Implementacija

V tem poglavju je predstavljena naša implementacija sistema za detekcijo in sledenje oznake na teoretični osnovi, ki je bila podana v prejšnjem poglavju. Najprej je opisana zasnova oznake, ki je osnova za delovanje detekcije in sledilnika. Oznaka je zasnovana s pomočjo orodja Adobe Illustrator. Nato sledi opis sistema kot celote, ki je v nadaljevanju razčlenjen na podpoglavje o detekcijskem algoritmu in podpoglavju o sledilnem algoritmu. V detekcijskem delu poglavja je podrobno opisana implementacija in delovanje detekcijskega algoritma. Za konec opišemo še implementacijo in delovanje sledilnega algoritma. Detekcijski in sledilni algoritem sta implementirana v jeziku Python z uporabo knjižnice Numpy in OpenCV.

3.1 Zasnova oznake

Zasnova oznake je prvi korak pri implementaciji sistema za avtomatsko detekcijo in sledenje določeni oznaki. Oznako je potrebno zasnovati, da se lahko pri izvajanju algoritma nad videoposnetkom sklicujemo na oznako v sličici in s pomočjo oznake določamo njeno pozicijo. Za potrebe naše magistrske naloge smo zasnovali krožno oznako in jo prilagodili glede na zahtevane pogoje. Za namene testiranja smo se odločili uporabiti dve oznaki, ki se razlikujeta v barvi ozadja in barvi krožnic ter debelini posameznih krožnic. Zasnovani

(a) Oznaka z belim ozadjem in črnimi krožnicami.

(b) Oznaka s črnim ozadjem in belimi krožnicami.

Slika 3.1: Zasnovani oznaki.

krožni oznaki sta prikazani na sliki 3.1. Pri zasnovi oznake želimo doseči preprostost, enoličnost, razločnost med oznako in okolico, v katero je postavljena in hranjenje informacij, ki jih sistem potrebuje za pravilno delovanje.

Oznaka je preprostih oblik in vzorcev, kar omogoča enostavno detekcijo. Vzorci v oznaki določajo enoličnost, da oznake ni mogoče zamenjati za kakšen drugi objekt ali regijo v sliki. Razločnost med oznako in okolico je ključnega pomena za detekcijo in sledenje oznake. Oznaka se mora razlikovati v barvi in obliki od ozadja tako, da ustvarimo čim bolj razločno mejo med oznako in okolico. Z zasnovo oznake določene oblike se definira tudi informacije, ki jih je mogoče pridobiti s pomočjo algoritma ter jih uporabiti za določitev lokacije in razdalje kamere od oznake. Potrebno se je zavedati, da je sistem za avtomatsko detekcijo in sledenje oznaki namenjen za delovanje v naravi, kar pomeni, da nimamo na razpolago nadzorovanega okolja s prilagojenimi razmerami. Razmere v naravi so glavni faktor, ki odločajo, kakšno oznako bomo potrebovali za uspešno delovanje algoritma. Pomembni faktorji so svetloba, barva in oblike, ki se pojavljajo v okolici oznake. Pri svetlobi se pozna odboj svetlobe od oznake, kar lahko povzroča bleščanje. Barve med oznako in okolico morajo biti čim bolj kontrastne in oznaka naj ima oblikovne lastnosti, ki jih v okolici oznake ni oziroma se malokrat pojavljajo.

Naša zasnovana oznaka vsebuje več koncentričnih krožnic. Pomembna lastnost krožne oznake je invariantnost na rotacijo in translacijo. Poleg tega se krožnica pri preslikavi iz planarne v neplanarno ravnino pretvori v elipso in s tem ne bistveno spremeni ali oteži detekcijo oznake. Pri zasnovi krožne oznake smo se osredotočili na tri lastnosti oznake, ki bistveno vplivajo na natančnost in uspešnost detekcijskega algoritma. Prva lastnost je število krožnic v oznaki. Z večjim številom krožnic omogočimo večjo verjetnost zagotovitve, da je detekcijski algoritem detektiral pravo oznako. V naravi se tudi najde elipsaste oblike vendar ponavadi niso sestavljene iz velikega števila koncentričnih krožnic oz. elips, kar nam ponuja krožna oznaka. Z eksperimentiranjem smo določili, da je glede na velikost oznake najbolj primerno uporabiti 3 krožnice, ker so v nasprotnem primeru krožnice preveč skupaj in se izgubi razločnost med posameznimi krožnicami pri večjih razdaljah.

Druga lastnost, ki vpliva na natančnost in učinkovitost detekcije oznake, je debelina krožnic v oznaki. Debelina in število krožnic sta soodvisni lastnosti, ker v primeru tanjših krožnic je mogoče na enako velikost oznake narisati več krožnic, kot če so krožnice debelejše. Pri poskušanju detekcije različnih oznak z različno debelino krožnic smo prišli do zaključka, da zaradi odboja svetlobe pride do učinka, ki se imenuje cvetenje, kar vpliva na jasnost oznake. Cvetenje (ang. bloom) je učinek, pri katerem kamera oz. človeško oko zazna iluzijo raztezanja močne svetlobe čez rob osvetljenega telesa na sceni. Za zmanjšanje učinka cvetenja je potrebno uporabiti debelejše krožnice, kar posledično vpliva na boljšo razločnost med posameznimi krožnicami. Potrebno je poudariti, da to velja le za oznake s svetlim ozadjem in temnimi krožnicami. V našem primeru gre za oznako z belim ozadjem in črnimi krožnicami.

Zadnja lastnost je barva oznake, ki mora biti prilagojena glede na okolje, v katerem se oznaka nahaja in na kontrastnost barv same oznake, da je vzorec enostavno detektiran. Osnovna zasnovana oznaka vsebuje črno barvo, ki predstavlja krožnice in belo barvo, ki predstavlja ozadje oznake. Ustvarjeni kontrast nam omogoča dobro razločnost med krožnicami in belim ozadjem.

Poleg tega sta črna in bela barva dobri tudi za razločitev med oznako in okolico, v kateri se oznaka nahaja. V naravi prevladujejo zeleni odtenki za travnate površine in gozdove, modri odtenki predstavljajo vodo in rjavi ter sivi odtenki predstavljajo zemljo in kamenje. Za vse naštete barve sta najbolj univerzalni kontrastni barvi črna in bela. Poleg belo-črne oznake smo zasnovali tudi črno-belo oznako, ki ima za ozadje črno barvo, medtem ko so krožnice narisane z belo barvo. Oznaka z črnim ozadjem in belimi krožnicami je bila zasnovana zaradi prej omenjenega cvetenja v primeru močne svetlobe. Bela barva odbija svetlobo veliko bolj kot črna, zato smo razvili tudi oznako s črnim ozadjem, ki vpija svetlobo in tako zmanjša jakost bleščanja ter cvetenja.

Ne glede na to, da smo za potrebe naše magistrske uporabili črno in belo barvo ter krožnice za zasnovo oznake, je vedno potrebno analizirati razmere, v katerih naj bi sistem deloval. Odvisno od razmer, se je potem potrebno odločiti, kakšno oznako bomo zasnovali. Zasnovati oznako, ki bi bila kos vsem razmeram ni mogoče, vendar lahko z majhnimi prilagoditvami izboljšamo delovanje detekcije in sledenja na raven, ki zadostuje potrebam sistema.

3.2 Celoten sistem za detekcijo in sledenje oznake

Algoritem celotnega sistema je razdeljen na tri dele. Prvi del vključuje detekcijski algoritem, drugi del poskrbi za sledenje oznake in tretji del določa zanesljivost sledenja in v primeru nezanesljivosti aktivira ponovno izvedbo detekcije oznake. Psevdokoda algoritma je zapisana v algoritem 1. Najprej je potrebno inicializirati strukturo za hranjenje stanja detektorja in strukturo za hranjenje stanja sledilnika skozi celoten proces. Nato je potrebno inicializirati podatke o videoposnetku in pripraviti videoposnetek za branje. Ko je vse pripravljeno, se v zanki začne izvajati celoten algoritem.

Najprej se prebere začetno sličico videoposnetka in se vsakič preveri, ali smo prišli do konca sličic videoposnetka. Če je algoritem prebral vse sličice,

Algoritem 1 Algoritem za detekcijo in sledenje znane oznake.

```
1: InitDetectionState[DS(0)]
2: InitTrackingState[TS(0)]
3: InitVideoData[Data]
4: while frame = Data.sequence.read() not empty do
5:    gray ← rgb2gray(frame)
6:    hsv_frame ← rgb2hsv(frame)
7:    if DS(i).markerFound equals false then // če detektor ni detektiral oznake
8:       edgeImg ← canny(gray)
9:       DS(i) ← detectMarker(edgeImg, hsv_frame, DS(i))
10:   end if
11:   if DS(i).markerFound equals true then // če je detektor detektiral oznako
12:      if TS(i).needToInitT equals true then // če je potrebno inicializirati sledilnik
13:         TS(i) ← AdditionalInitTrackingState[hsv_frame, TS(0), DS(i)]
14:         TS(i).needToInitT ← false
15:      end if
16:      TS(i) ← trackMarker(hsv_frame, TS(i))
17:      if TS(i).notReliable equals true then // če je rezultat sledenja nezanesljiv
18:         DS(i).markerFound ← false
19:         TS(i).needToInitT ← true
20:      end if
21:   end if
22:   i ← i + 1
23: end while
```

se program zaključi. Vsako prebrano sličico pretvorimo v sivinsko sliko ter sliko v barvnem prostoru HSV, ker sta pretvorjeni sliki potrebni v naslednjih korakih. Prvi pogoj preveri, če detektor ni detektiral oznake. V prvi iteraciji zanke je pogoj vedno izpolnjen in se najprej pridobi sliko robov s pomočjo Canny algoritma, ki je podrobneje opisan v podpoglavju 3.2.1. Pridobljena slika robov je nato podana kot parameter v funkcijo za detekcijo oznake skupaj s sliko v barvnem prostoru HSV ter stanjem detektorja. Detekcijski algoritem je namenjen iskanju vzorcev v sliki robov, ki se ujemajo s podano oznako. V našem primeru so to elipse, ker imamo krožno oznako. Če je detektor našel oznako se poleg posodobljenega stanja detektorja pridobi tudi informacije o poziciji in velikosti oznake. Podroben opis delovanja detekcijskega algoritma je v podpoglavju 3.3. V primeru, da je bil detektor neuspešen pri detekciji, se enak postopek ponovi na naslednji sličici videoposnetka.

Če je detektor našel oznako, kar preveri pogoj po izvedeni detekciji, se začne del, ki zajema sledenje oznake. Če se sledenje prvič izvaja, je potrebno posodobiti stanje s stanjem, ki ga vsebuje detektor in tako prenesti informacije o lokaciji in velikosti oznake. Ta korak je potreben le pred prvim izvajanjem sledenja in v primeru, ko sledilnik vrne nezanesljiv rezultat oznake. Sledilnik izvede sledenje in posodobi stanje sledilnika glede na uspešnost sledenja. Sledilni algoritem je podrobneje opisan v 3.4.

Če je rezultat sledenja zanesljiv, se prebere naslednja sličica in ponovi postopek sledenja brez izvajanja detekcije. V primeru, da je sledilnik vrnil nezanesljiv rezultat, se ponovno izvede detekcija na naslednji sličici in se tako izvaja celoten algoritem do konca videoposnetka.

3.2.1 Cannyjev algoritem za detekcijo robov

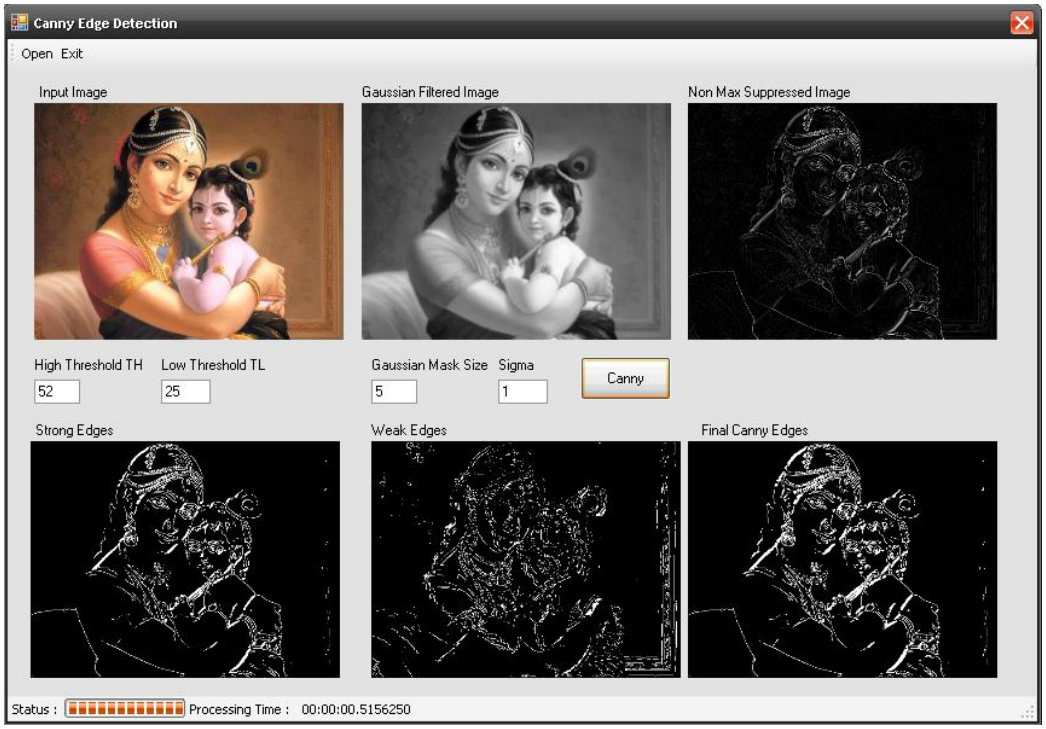
Canny algoritem omogoča detekcijo robov na sivinski sliki. Posamezni koraki algoritma so prikazani na sliki 3.2. Algoritem deluje tako, da sprejme sivinsko sliko skupaj s parametroma, ki predstavljata spodnji in zgornji prag za filtriranje šibkih in močnih robov. V prvem koraku algoritem zmanjša šum v sliki s pomočjo Gaussovega filtra. Naslednji korak poskrbi za izračun

gradientne slike, ki prikazuje smer in moč posameznega roba v sliki. Moč roba se izračuna po enačbi (3.1), medtem ko se smer roba izračuna po enačbi (3.2), pri čemer sta G_x in G_y odvoda filtrirane slike po osi x in osi y.

$$G = \sqrt{G_x^2 + G_y^2} \tag{3.1}$$

$$\theta = atan2(G_x, G_y) \tag{3.2}$$

V tretjem koraku nastopi tehnika ožanja robov (Non-maximum-suppression). Ker se v gradientni sliki robovi raztezajo čez več slikovnih elementov, je potrebno s tehniko ožanja robov vsak rob zožiti. Ožanje robov deluje tako, da v vsakem robu poiščemo lokalni maksimum gradienta in ohranimo le tega. Ostale gradiente zavržemo in tako se ožanje izvede nad vsemi gradienti v gradientni sliki. Četrti korak je namenjen ločevanju pravih od nepravih robov.



Slika 3.2: Prikaz vmesnih korakov iskanja robov [50].

V prejšnjem koraku dobimo natančnejšo sliko pravih robov v sliki, vendar

so zaradi šuma in barvnih variacij nekateri robovi napačni. Da bi izločili napačne robove, lahko uporabimo metodo pragov, ki poskrbi, da izloči piksle z majhnim gradientom in ohrani tiste z visokim gradientom. Za ta način filtriranja sta potrebna prej omenjena pragova. Zgornji prag določa mejo, pri kateri se slikovni element sprejme kot močan rob, če je njegov gradient večji od tega praga. Če je gradient piksla manjši od zgornjega praga in večji od spodnjega praga, se pikel uvrsti kot šibek rob. Gradienti pikslov, ki so manjši od spodnjega praga, se zavržejo.

V zadnjem koraku se piksle z močnimi robovi vključi v končno robno sliko, medtem ko je piksle s šibkimi robovi potrebno še dodatno preveriti ali jih vključiti v končno robno sliko ali ne. Metoda, ki preveri šibke robove, temelji na preverjanju povezanosti šibkih robov z močnimi robovi. Vsak piksel, ki predstavlja šibek rob, se vzame in preveri sosednje piksle, če kateri od njih vsebuje močan rob. V primeru povezanosti šibkega roba z močnim robom se šibek rob ohrani in vključi v končno robno sliko. Ostale piksle, ki ne izpolnjejo pogojev povezanosti šibkega roba z enim ali več močnimi robovi, se zavrže [24].

3.3 Detekcijski algoritem

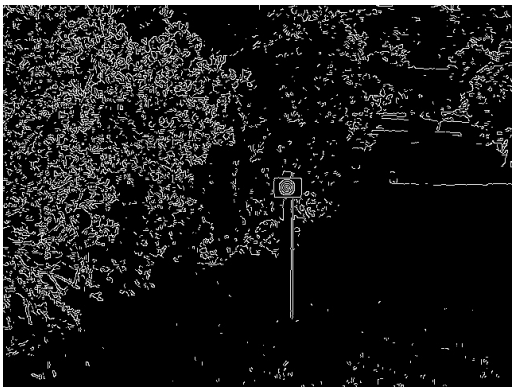
Naloga detekcijskega algoritma je detektirati zasnovano krožno oznako na posamezni sličici videoposnetka in izluščiti informacije o lokaciji in velikosti oznake. Za potrebe detekcije smo razvili algoritem, ki omogoča detektirati elipse v sliki in tako prepoznati krožno oznako. Glavni koraki algoritma so zapisani v psevdokodi algoritma 2. Detekcijski algoritem je zasnovan kot funkcija, ki sprejme za vhod sliko robov, sliko v barvnem prostoru HSV in strukturo stanja detektorja. Prvi korak detekcijskega algoritma je poiskati obrise v sliki robov. Iz pridobljenih obrisov se izlušči 5 najboljših kandidatov, ki so potencialni predstavniki oznake. Vsakega od kandidatov za oznako se

Algoritem 2 Algoritem za detekcijo oznake.

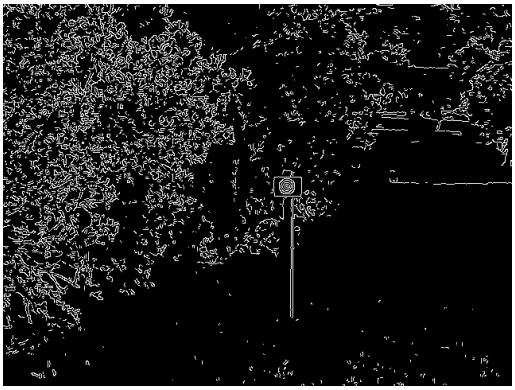
1: $edgeImg, hsvImg, DS(i)$
2: $data \leftarrow DS(i)$ // uporabi trenutno stanje detektorja.
3: $contours, hierarchy \leftarrow findContours(edgeImg)$
4: $topCandidates \leftarrow findTopContourCandidates(contours, hierarchy)$
5: $bestMatch \leftarrow selectBestMatch(topCandidates)$
6: $DS(i) \leftarrow updateDetectionState(data)$
7: **return** $DS(i)$ // vrni novo stanje detektorja

nato individualno oceni in izbere najverjetnejšega. Izbranega kandidata se potrdi kot oznako in s pridobljeno strukturo izlušči informacije o lokaciji in velikosti oznake. Z novimi informacijami se novo posodobljeno stanje detektorja vrne kot rezultat funkcije. Vsak od korakov detekcijskega algoritma je natančneje razložen v naslednjih podpoglavjih in prikazan na sliki 3.3.

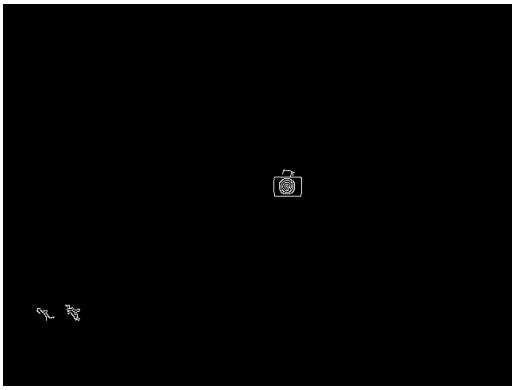
(a) Vhodna slika.



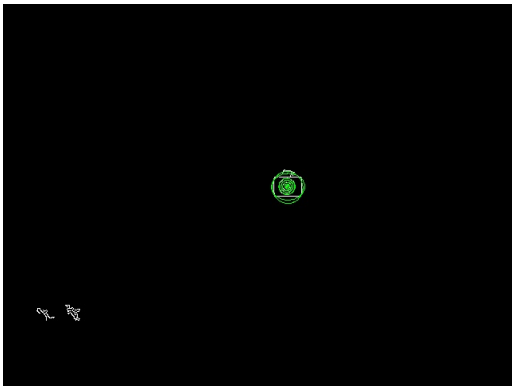
(b) Slika robov.



(c) Slika obrisov.



(d) Najboljših 5 skupin obrisov.



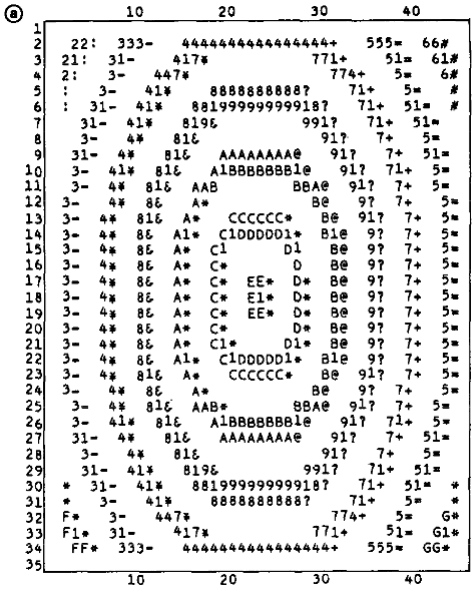
(e) Najboljša skupina obrisov in prilegajoče se elipse.

(f) Rezultat detekcije.

Slika 3.3: Glavni koraki celotnega postopka za detekcijo oznake.

3.3.1 Iskanje obrisov

Prvi korak detekcijskega algoritma za detekcijo krožne oznake je iskanje obrisov v sliki robov s pomočjo funkcije *findContours*, ki je del knjižnice OpenCV. Funkcija *findContours* za iskanje obrisov deluje nad binarno sliko. V zanki se sprehodi čez vse vrstice od zgoraj navzdol in išče neničelni piksel. Vsak neničelni piksel predstavlja rob, po katerem se algoritem sprehodi tako, da mu sledi po neprekinjeni sledi enic, dokler ne naleti na ničlo. Vsak obris je označen z lastno identifikacijsko številko, da se prepozna že najdene obrise. Obrisi predstavljeni z lastnimi identifikacijskimi številkami so prikazani na sliki 3.4. Po vsakem najdenem obrisu se postopek ponovno nadaljuje od loka-



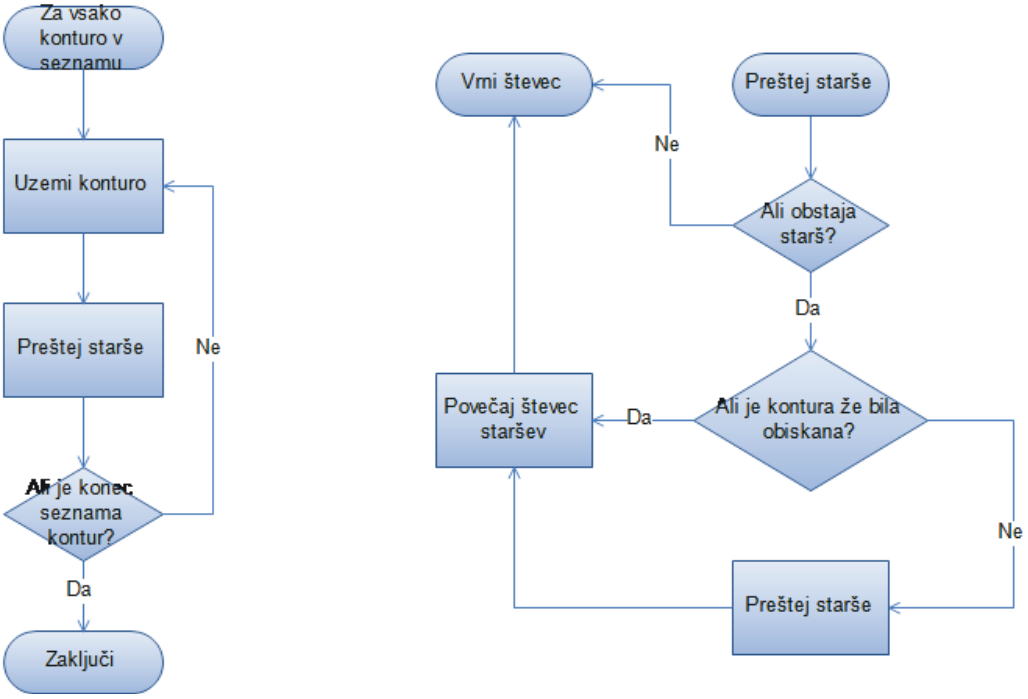
Slika 3.4: Rezultat iskanja obrisov v binarni sliki [51].

cije, kjer se je začelo sledenje robu. Algoritem se zaključi v desnem spodnjem kotu slike in kot rezultat vrne seznam obrisov in seznam, ki hrani podatke o hierarhiji najdenih obrisov. Rezultat algoritma je prikazan na sliki 3.3 (c). Podroben opis algoritma je opisan v [51].

3.3.2 Iskanje najboljših kandidatov

Za iskanje najboljših kandidatov oznake sta zadolženi dve funkciji. Prva funkcija uporabi rezultat iz prejšnjega koraka tako, da vsakemu obrisu prešteje število starševskih obrisov. Cilj je pridobiti informacije o številu vgnezdenih obrisov, ker našo oznako sestavljajo vgnezdene elipse in na ta način pridobimo glavne kandidate za določitev lokacije oznake.

V prvem delu iskanja najboljših kandidatov se v zanki, ki se sprehodi čez vse obrise, pokliče nad vsakim obrisom rekurzija, ki prešteje vse starše trenutno procesiranega obrisa. Delovanje rekurzije je prikazano z desnim diagramom na sliki 3.5, medtem ko je zanka za sprehod čez vse obrise prikazana na levem diagramu.



Slika 3.5: Levo zanka, ki se sprehodi čez seznam obrisov v sliki, desno rekurzivna funkcija za štetje staršev obrisov.

Rekurzija vzame vsak obris in s pomočjo podatkov o hierarhiji vgnezdenih obrisov prešteje vse starše določenega obrisa. Za optimizacijo procesa

skrbi pogoj, ki v primeru, da je bil starš že obiskan, ni potrebno ponovnega štetja obrisov od obiskanega starša naprej, ampak se prišteje že pridobljeno število staršev, ki ga hrani obiskani starš. Rezultati rekurzije se shranijo v seznam objektov, ki vsebujejo podatka o številu staršev in seznam vgnezdenih obrisov. Seznam objektov se sortira sproti, ker se rezultate rekurzije vstavlja sortirano v seznam in ohranja le najboljših pet kandidatov. Najboljši kandidati so tisti, ki vsebujejo največje število vgnezdenih obrisov. Rezultat iskanja najboljših pet kandidatov za oznako je predstavljen na sliki 3.3 (d). Izbira najboljšega kandidata za oznako je opisana v naslednjem podpoglavju.

3.3.3 Izbira najboljšega kandidata

Iz pridobljenih najboljših 5 kandidatov v prejšnjem koraku, je naloga tega koraka izluščiti najboljšega kandidata za oznako. Postopek, ki je prikazan v psevdokodi algoritma 3, je sestavljen iz dveh glavnih zank. Zunanja zanka se sprehodi čez vseh 5 kandidatov, medtem ko notranja zanka poskrbi za sprehod čez vse obrise posameznega kandidata. V notranji zanki se najprej s pomočjo funkcije *fitEllipse*, ki je del knjižnice OpenCV, vsakemu obrisu izračuna najbolj prilegajočo elipso. Opis delovanja funkcije *fitEllipse* za iskanje najbolj prilegajoče elipse je opisan v [52].

Vsako elipso se preveri ali ustreza kriterijem, ki so določeni empirično. Kriterijem, ki jim mora biti zadoščeno, so velikost elipse, pozicija elipse in razmerje horizontalnega polmera z vertikalnim.

$$diamX < imgW/2 \ \vee \ diamY < imgH/2 \tag{3.3}$$

$$x > \epsilon \ \vee \ x < (imgWidth - \epsilon) \tag{3.4}$$

$$y > \epsilon \ \vee \ y < (imgHeight - \epsilon) \tag{3.5}$$

$$diamR < highT \ \vee \ diamR > lowT \tag{3.6}$$

Elipsa je zavržena, če je njen premer prevelik, za kar poskrbi kriterij z enačbo (3.3). Kriterija (3.4) in (3.5) poskrbita, da se filtrirajo elipse, ki imajo točko centra izven ϵ območja na robu slike. Zadnji kriterij (3.6) izloči elipse, ki

Algoritem 3 Detekcijski algoritem - iskanje najboljšega kandidata za oznako.

for $contoursintopContoursCandidates$ **do**
  for $contourincontours$ **do**
    $ellipse \leftarrow fitEllipse(contour)$
    $checkEllipseAnomalies(ellipse)$
    $ellipseGroup.append(ellipse)$
  end for
  $ellipsesData \leftarrow calcCenterPointAndRadiusOfEllipses(ellipseGroup)$
  $currentRegionHist \leftarrow determineMarkerRegionHist(ellipsesData)$
  $matchScore \leftarrow compareHists(currentRegionHist, markerTemplateHist)$
  $bestMatchRegion \leftarrow keepBestMatch(matchScore)$
end for
$filterOutlierEllipses(bestMatchRegion.ellipses)$
$DS(i) \leftarrow updateDetectorState(bestMatchRegion)$
return $DS(i)$

imajo razmerje med vertikalnim in horizontalnim premerom večje oz. manjše od določenega praga.

Vse elipse enega kandidata se shrani v en seznam, iz katerega nato izluščimo informacije o povprečnem centru in povprečnem premeru elips. Informacije o centru in premeru elips se uporabi za določitev regije oznake. Pridobljena regija je lokacija, na kateri je možnost, da se nahaja iskana oznaka. Za določitev zanesljivosti morebitne regije oznake se s pomočjo primerjave ujemanja histogramov opredeli podobnost med histogramoma predloge oznake in detektirane regije oznake. Izbira najboljšega kandidata za regijo oznake je določena glede na nivo ujemanja predloge oznake z najdeno regijo oznake. Najboljša regija oznake je tista, ki doseže najvišje ujemanja med histogramoma. Primer najboljše regije je prikazan na sliki 3.3 (e).

V zadnji fazi se najboljšo regijo oznake, predstavljeno s skupino elips, filtrira tako, da se izloči vse elipse, ki se od povprečnega centra razlikujejo za določen ϵ merjeno v slikovnih elementih.

$$|x - \bar{x}| > \epsilon \ \wedge \ |y - \bar{y}| > \epsilon \tag{3.7}$$

Enačba (3.7) prikazuje filtrirna pogoja za posamezno elipso. Po filtriranju je

potrebno ponovno izračunati center regije oznake in območje okrog centra, da se shrani lokacijo oznake in posodobi stanje detekcijskega algoritma. Končni rezultat detekcije je prikazan na sliki 3.3 (f).

3.4 Sledilni algoritem

Sledilni algoritem je podobno kot detekcijski algoritem namenjen iskanju oznake v podani sličici videoposnetka, vendar se razlikujeta po dveh ključnih lastnostih. Sledilni algoritem uporablja drugačno metodo za lociranje iskane oznake in zaradi predpostavke, da je sprememba pozicije oznake med dvema zaporednima sličicama majhna, se izvajanje sledilnega algoritma omeji le na del celotne slike. Sledilni algoritmi so zelo koristni zaradi hitrejšega procesiranja in lociranja želene oznake. Za potrebe naše magistrske naloge smo razvili algoritem za sledenje krožni oznaki, ki temelji na osnovi algoritma Sprotno prilagodljiv povprečni premik (CamShift - Continuously adaptive Mean-Shift).

Implementirani sledilni algoritem je del celotnega sistema, ki se aktivira po uspešni detekciji krožne oznake s pomočjo detekcijskega algoritma opisanega v poglavju 3.3. Sledilni algoritem je razdeljen na dve glavni funkciji. Prva funkcija je namenjena inicializaciji parametrov, ki so potrebni za pravilno delovanje sledilnega algoritma. Inicializacija je predstavljena v psevdokodi algoritma 4.

Algoritem 4 Inicializacija sledilnega algoritma.

1: $hsvImg, TS(i), DS(i)$
2: $roi \leftarrow initMarkerRegion(hsvImg, centerP, radius)$
3: $roi\_hist \leftarrow calcHist(roi, bins, ranges)$
4: $window \leftarrow initTrackingWindow(hsvImg, centerP, radius, offset)$
5: $TS(i) \leftarrow updateTrackerState(roi, window, roi_hist)$
6: **return** $TS(i)$

V inicializacijski fazi sledenja, ki se izvede le prvič, ko vstopimo v sledilni del sistema, se najprej iz detekcijskega stanja izlušči vse potrebne informacije

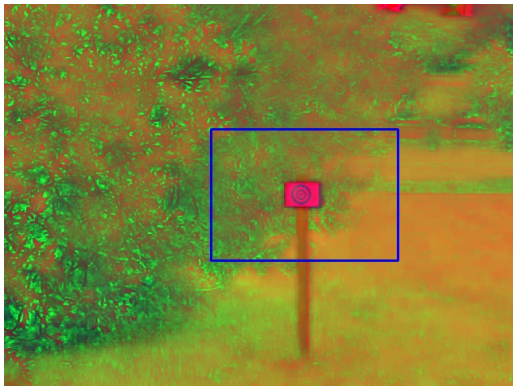
za določitev začetne regije oznake. Nato se izračuna barvni histogram regije oznake, ki nam služi kot referenca za sledenje oznaki v posamezni sličici videoposnetka. Preden začnemo z dejanskim sledenjem, se pripravi še okno za sledenje, ki je manjše od celotne slike za določen zamik. Zadnji korak inicializacije je namenjen posodobitvi stanja sledilnika s pridobljenimi informacijami o regiji oznake, histogramu oznake in oknu za sledenje oznake. S tem je inicializacije konec in algoritem za sledenje lahko vstopi v fazo sledenja z novo posodobljenim sledilnim stanjem.

Sledilna faza algoritma se začne z izračunom projekcije histograma oznake na sledilno okno. Sledilna faza je predstavljena v psevdokodi algoritma 5. Projekcijska slika histograma nam vrne verjetnostno sliko pozicije oznake v trenutno procesirani sličici videoposnetka, ki jo uporabimo kot vhod v sledilni algoritem Sprotno prilagodlijv povprečni premik (CamShift). CamShift nam vrne morebitno regijo oznake na določeni lokaciji znotraj sledilnega okna.

Algoritem 5 Sledilni algoritem.

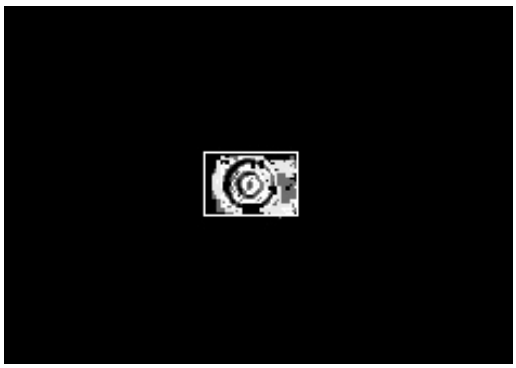
1: $hsvImg, TS(i)$
2: $dst \leftarrow calcBackProject(hsvImg, roiHist, ranges)$
3: $res, roi \leftarrow CamShift(dst, roi, term_crit)$
4: $trackData \leftarrow proccesTrackingData(res, roi)$
5: **if** $trackData.reliable$ equals **true then** // če so rezultati sledenja zanesljivi
6: $\quad TS(i) \leftarrow updateTrackerState(trackData)$
7: **else**
8: $\quad TS(i).reliable \leftarrow$ **false**
9: **end if**
10: **return** TS(i)

Vrnjeno regijo se nato preveri, ali ustreza pogojem iskane oznake, določi zanesljivost regije oznake in obdela podatke o novi lokaciji oznake za naslednjo sličico. Če je vrnjena regija oznake zanesljiva, se posodobi stanje sledilnika in vrne novo stanje sledilnika, ki se ga uporabi za sledenje v naslednji sličici videoposnetka. V primeru, da je rezultat sledenja nezanesljiv, se rezultat zavrže in sproži detekcijski algoritem za ponovno lokalizacijo oznake. Postopek sledenja se po ponovni detekciji ponovi od začetka. Najprej se



(a) Vhodna slika z označenim sledilnim oknom.

(b) Projekcijska slika.



(c) Rezultat sledilnika nad projekcijsko sliko.

(d) Rezultat sledenja.

Slika 3.6: Glavni koraki celotnega postopka za sledenje oznake.

ponovno inicializira sledilnik in nato izvede sledenje. Sledilni algoritem je podrobneje opisan v naslednjih podpoglavjih in po stopnjah predstavljen na sklopu slik 3.6.

3.4.1 Inicializacija sledilnika

Kot smo že omenili, je inicializacija sledilnika potrebna le pred začetkom prvega izvajanja sledenja. Najprej je potrebno iz stanja detekcijskega algoritma izluščiti informacije o lokaciji in velikosti detektirane krožne oznake, ki je predstavljena z njenim centrom, višino in širino. S pomočjo lokacije in

velikosti oznake se pripravi regijo oznake in iz nje se izračuna barvni histogram, ki služi kot sklic za določitev nadaljnjih lokacij oznake v sličicah videoposnetka. Pri določanju regije oznake se vedno preveri ustreznost velikosti regije in ali regija presega meje, ki določajo okvir procesirane slike. Pogoja za preseg velikosti slike sta zapisana z enačbama (3.8) in (3.9). Ustreznost velikosti regije oznake pa določa enačba (3.10).

$$x1, x2 > 1 \ \wedge x1, x2 < imgWidth \tag{3.8}$$

$$y1, y2 > 1 \ \wedge y1, y2 < imgHeight \tag{3.9}$$

$$|x2 - x1| > \epsilon \ \wedge |y2 - y1| > \epsilon \tag{3.10}$$

Preverjanje ustreznosti regije se uporablja tudi v sledilnem delu algoritma, kjer je potrebno vsakič preveriti novo pridobljeno lokacijo oznake. Za pridobitev barvnega histograma krožne oznake se uporablja funkcija *calcHist*, ki sprejme kot vhod sliko regije oznake, število odtenkov (bins) v posameznem barvnem kanalu in razpon posameznega barvnega kanala. Funkcija *calcHist* je podrobneje opisana v poglavju 3.4.7.

V naslednjem koraku inicializacije je potrebno določiti še okno, v katerem bo sledilnik iskal določeno krožno oznako. Okno je določeno tako, da razširimo velikost regije oznake za 4 kratnik v horizontalni in vertikalni smeri slike. Zadnji korak predstavlja posodobitev vhodnega stanja sledilnega algoritma z vsemi novo pridobljenimi informacijami tako, da bo lahko algoritem pravilno deloval. Poleg opisanih informacij je potrebno dodati še nekaj konstant, ki pripomorejo k pravilnemu iskanju in posodabljanju nove lokacije oznake. Ko je stanje sledilnika posodobljeno, algoritem prične z izvajanjem sledenja.

3.4.2 Sledenje

Sledenje se začne z vstopom v sledilno funkcijo, ki sprejme dva parametra. Prvi parameter je trenutno procesirana sličica, ki je pretvorjena v barvni prostor HSV. Drugi parameter predstavlja trenutno stanje sledilnega algoritma,

iz katerega vzamemo parametre za pravilno delovanje sledenja. Parametri, ki jih predpripravimo so:

- širina in višina slike,
- histogram predloge oznake,
- kriterij za prekinitev izvajanja funkcije *CamShift*,
- regijo oznake,
- objekt Kalman,
- koordinate sledilnega okna in
- pet konstant za pravilno posodabljanje in prilagajanje algoritma.

Najprej je potrebno s pomočjo koordinat sledilnega okna izrezati iz vhodne slike sledilno okno, v katerem nato poteka sledenje. Sledilno okno je prikazano na sliki 3.6 (a). Sledilno okno in histogram predloge oznake se uporabi za izračun projekcijske slike histograma na sliko sledilnega okna. Projekcijska slika nam poda informacijo, kje na sliki se najverjetneje nahaja oznaka, ki je predstavljena z barvnim histogramom.

Za izračun projekcjiske slike poskrbi funkcija *calcBackProject*, ki je del knjižnice OpenCV. Za izračun projekcijske slike smo uporabili le kanala barvni odtenek in nasičenost iz barvnega prostora HSV, ker se tako najbolje razloči barvo ozadja od barve oznake. Vhodni parametri v funkcijo *calcBackProject* so kanal barvni odtenek (hue) in kanal nasičenost (saturation), HS barvni histogram predloge oznake in parametra za razpon vrednosti posameznega barvnega kanala. V rezultatu projekcijske funkcije dobimo projekcijsko sliko, ki predstavlja najverjetnejšo lokacijo iskane oznake znotraj sledilnega okna. Slika 3.6 (b) predstavlja izhod projekcijske funkcije. Podroben opis delovanja funkcije *calcBackProject* je v podpoglavju 3.4.3.

Projekcijsko sliko dodatno upragujemo za izločitev šuma in jo nato podamo kot parameter v sledilno funkcijo *CamShift*. Sledilna funkcija poleg

projekcijske slike prejme zadnjo lokacijo regije oznake in zaključitveni pogoj funkcije sledenja. V našem primeru je to 10 iteracij ali če se regija oznake, ki se sprehaja čez projekcijsko sliko, premakne le za 1 slikovni element. Ko je eden od obeh kriterijev izpolnjen, funkcija vrne rezultat, ki predstavlja regijo oznake v obliki pravokotnika. Na sliki 3.6 (c) je predstavljen rezultat sledenja nad projekcijsko sliko. Funkcija *CamShift* je podrobno opisana v poglavju 3.4.4.

Sedaj se začne zadnja faza, ki preveri, ali je bila regija oznake najdena in se nato glede na vrnjen rezultat vrne posodobljeno stanje sledilnega algoritma. Če regija oznake ni bila najdena, se algoritem zaključi s tem, da se zapiše v posodobljeno stanje, da je bilo sledenje izvedeno nezanesljivo. V primeru, da je bil rezultat funkcije *CamShift* uspešen, je potrebno določiti robne točke regije oznake, izračunati center regije, njeno širino in višino. Z informacijami o višini in širini se izračuna še razmerje med širino trenutne regije in širino prejšnje regije. Enako se določi tudi razmerje višine trenutne regije z višino prejšnje regije. Poleg razmerij je potrebno pridobiti še informacijo o razdalji med trenutno izračunanim centrom in predvidenim centrom, ki se ga določi s pomočjo Kalmanovega filtra.

Pridobljene informacije je potrebno spustiti skozi več filtrirnih pogojev, da lahko določimo zanesljivost rezultata. Prvi filter, ki je prikazan z enačbo (3.11), zavrne rezultat sledenja, če je razmerje višine ali širine preveliko oziroma premajhno. Pri drugem filtru se rezultat zavrže, če je izračunana razdalja med predvidenim centrom regije in izračunanim centrom prevelika. Pogoj za filtriranje z razdaljo je prikazan z enačbo (3.12). Če se rezultat ni zavrgel pri prvih dveh pogojih, se določi novo regijo oznake, ki se ravno tako preveri ali ustreza omejitvam tako kot v inicializacijski fazi v poglavju 3.4.1.

$$widthRatio, heightRatio < lowT \ \vee \ widthRatio, heightRatio > highT \tag{3.11}$$

$$distance > max(roi_{w}idth, roi_{h}eight) * scaleF \tag{3.12}$$

V primeru zavrženega rezultata sledenja se v stanje sledilnika zapiše, da je

rezultat nezanesljiv in algoritem vrne posodobljeno stanje sledilnika.

Rezultat, ki je ustrezal vsem pogojem za določitev zanesljivosti, se uporabi za posodobitev stanja sledilnika. Najprej se posodobi Kalmanov filter z novim centrom regije oznake ter napove center regije oznake na naslednji sličici. Delovanje Kalmanovega filtra je podrobneje opisano v poglavju 3.4.5. Stanje sledilnega algoritma se posodobi z:

- točko centra regije oznake,
- višino in širino regije oznake,
- robnimi točkami rotiranega pravokotnika regije oznake,
- regijo oznake,
- odmikom sledilnega okna od celotne slike in
- sledilnim oknom.

V zadnjem koraku se posodobi še histogram regije oznake, vendar le, če so izpolnjeni pogoji (3.13), (3.14) in (3.15). Prva dva pogoja določata, da se posodablja histogram le, če je razmerje širine/višine regije oznake in širine/višine slike večje od minimalnega razmerja in manjše od maksimalnega razmerja. Če sta izpolnjena prva dva pogoja, se histogram posodobi, ampak le na vsakih 10 sličic, ker ni potrebno, da se histogram posodablja iz sličice v sličico.

$$widthRatio, heightRatio < maxRatio \tag{3.13}$$

$$widthRatio, heightRatio > minRatio \tag{3.14}$$

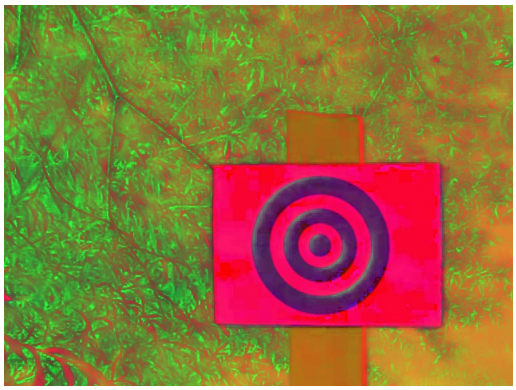
$$trackFrameCounter \tag{3.15}$$

Podroben opis posodabljanja oznake ob vseh izpolnjenih pogojih je predstavljen v poglavju 3.4.6. Ne glede na to, ali je bilo posodabljanje histograma oznake izvedeno ali ne, se za konec shrani v sledilno strukturo histogram oznake. Tako se sledenje na trenutni sličici videoposnetka zaključi in se

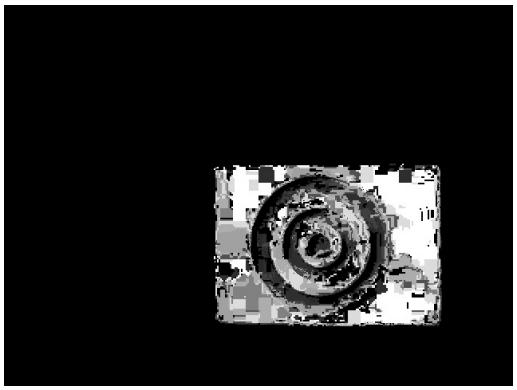
sledilna struktura prenese v naslednjo sledilno iteracijo, če je bilo sledenje uspešno izvedeno. V nasprotnem primeru se ponovno aktivira detektor, ki poskrbi za ponovno detekcijo oznake in postopek sledenja se začne znova.

3.4.3 Projekcijska funkcija

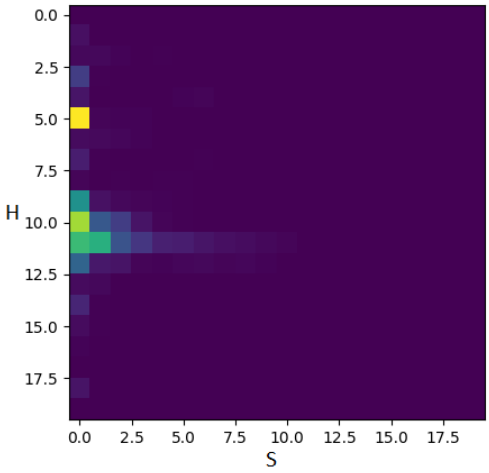
Funkcija *calcBackProject* se sprehodi skozi vsak slikovni element vhodne slike in iz vsakega slikovnega elementa izlušči informacije o vseh barvnih kanalih. Informacije o barvnih kanalih vsakega slikovnega elementa uporabi kot indeks v barvnem histogramu, ki je podan kot parameter funkciji.
Vsak indeks predstavlja lokacijo v N dimenzionalnem prostoru barvnega histograma, kjer se hrani informacija o količini posamezne vsebovane barve. Višja kot je količina vsebovane barve v barvnem histogramu, svetleje bo obarvan slikovni element v projekcijski sliki. Nižje vrednosti vsebovane barve obarvajo slikovni element s temnejšo barvo. Razpon barv v projekcijski sliki je od popolnoma črne barve, kar pomeni da barva ni vsebovana v histogramu, do bele barve, ki predstavlja maksimalno količino vsebovane barve v histogramu. Enak postopek se izvede za vsak slikovni element, dokler se ne sestavi celotna projekcijska slika. Primer projekcijske slike in pripadajoči histogram sta prikazana na sliki 3.7. Podrobno delovanje projekcijske funkcije je opisano v [53].



(a) Vhodna slika HSV.



(b) Projekcijska slika.

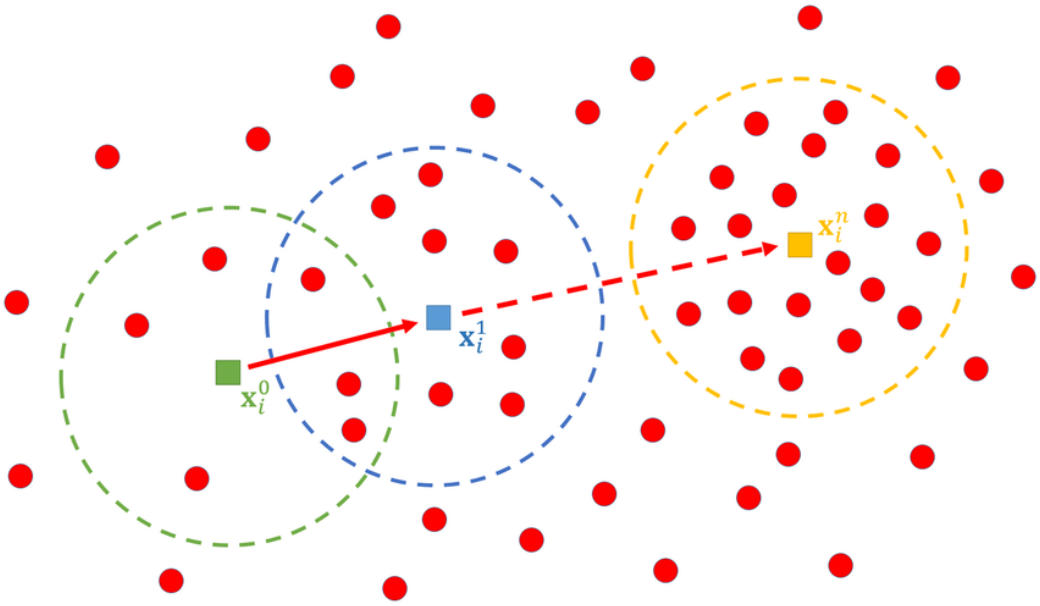


(c) Histogram oznake.

Slika 3.7: Projekcijska slika s pripadajočim histogramom in vhodno sliko.

3.4.4 CamShift

CamShift deluje po principu algoritma Povprečni premik (Mean-Shift), vendar z dodatkom, da se ob najdeni lokaciji iskane regije interesa prilagodi velikost okna, ki določa najdeno regijo, kar pa Povprečni premik ne omogoča in skozi celoten proces uporablja statično velikost okna. Algoritem Sprotno prilagodljivi povprečni premik (Cam-Shift) je razdeljen na dva dela. Prvi del predstavlja iskanje lokalnega maksimuma v projekcijski sliki in drugi del, ki najdeno regijo oriše s prilagoditvijo okna, ki je bilo podano kot vhodni paramater.



Slika 3.8: Premik okna regije interesa po izračunanih centrih mase [54].

Iskanje lokalnega maksimuma v projekcijski sliki se začne s postavitvijo okna regije interesa na začetno lokacijo. Obe informaciji sta podani kot vhodna parametra v funkcijo. Na trenutni lokaciji iskanja z določeno velikostjo okna se izračuna center mase, ki je predstavljen z verjetnostmi v projekcijski sliki. Algoritem pomakne okno v izračunani center mase in ponovno izračuna novi center mase. Primer pomikanja okna v lokalni maksimum je predstavljen na sliki 3.8. To počne dokler ne konvergira v lokalni maksimum oziroma

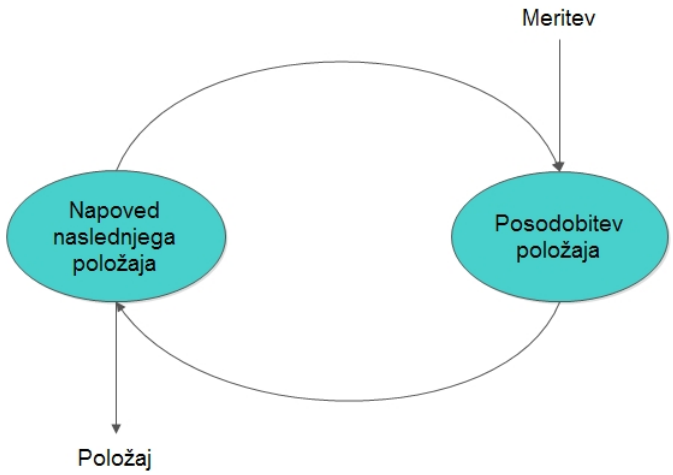
dokler se ne ustavi zaradi ustavitvenega pogoja.

Ustavitveni pogoj je podan kot vhod v funkcijo in določa, da se povprečni premik zaključi po N iteracijah ali če se iskalno okno ni več premaknilo za več kot n slikovnih elementov. V najdenem lokalnem maksimumu se izvede določitev nove regije oznake s pomočjo metode določanja najbolj prilegajoče elipse. Funkcija *CamShift* vrne rezultat v obliki rotiranega pravokotnika, ki predstavlja regijo interesa. Podrobno delovanje sledilnega algoritma CamShift je predstavljeno v [55].

3.4.5 Kalmanov filter

Kalmanov filter je metoda, ki se veliko uporablja pri napovedovanju nove lokacije sledenega objekta v sledilnih algoritmih. V osnovi je Kalmanov filter skupek matematičnih enačb, s katerimi je mogoče iz vhodnih meritev in učenjem napovedovati naslednjo vhodno meritev z določeno napako pod določenimi pogoji. Kalman filter ločimo na dva dela, pri katerem prvi del poskrbi za posodabljanje stanj na podlagi meritev in drugi del, ki poskrbi za napovedovanje meritve v času t+1. Kombinacija obeh delov metode določata pravilen položaj objekta v času t.

Delovanje filtra je odvisno od izbire modela gibanja. Model gibanja določa način gibanja sledenega objekta. Za gibanje s skoraj konstantno hitrostjo se uporablja model konstantne hitrosti, kateremu se doda šum, ki ustvarja dinamiko gibanja in bolje zajame napoved, saj je gibanje v naravi predvidljivo le z določeno stopnjo zanesljivosti. Za šum se ponavadi uporablja Gaussovo porazdelitev z določenim faktorjem σ. Na sliki 3.9 je predstavljen proces določanja položaja objekta na sliki s Kalmanovim filtrom. Napovedovalni del napove naslednji položaj s pomočjo prejšnjega stanja in kovariančno napako. Pri meritvenem delu se najprej izračuna Kalmanov doprinos (ang. Kalman gain) in nato s trenutno meritvijo določi v kombinaciji z napovedanim položajem nov točen položaj objekta na sliki. Poleg tega se posodobi kovariančno napako. Podrobnosti o Kalmanovem filtru so opisane v [56] in [13].

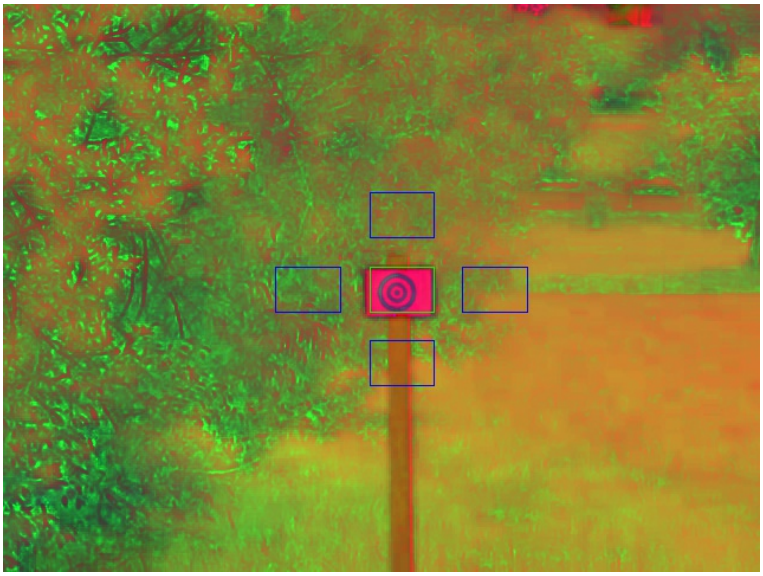


Slika 3.9: Proces določanja položaja s Kalmanovim filtrom.

3.4.6 Posodobitev histograma

Posodobitev histograma regije oznake je ključnega pomena, da sledilni algoritem deluje, saj je zaradi sprememb velikosti in barve regije oznake sledenje podvrženo napakam. Da bi se izognili napakam sledenja, je potrebno posodabljati predlogo barvnega histograma, ki predstavlja regijo oznake. Za posodobitev barvnega histograma so potrebne informacije regije oznake ter vrednost odmika, ki je namenjen za določitev ozadja okoli oznake. Najprej se izračuna histogram regije na način predstavljen v poglavju 3.4.7. Zaradi delov regije oznake, ki ne pripadajo oznaki, ampak ozadju, je potrebno izračunani histogram filtrirati s histogrami ozadja, da dobimo le barvne odtenke in nasičenosti, ki pripadajo oznaki.

Histogram oznake se filtrira tako, da se vzame del slike levo, zgoraj, desno in pod regijo oznake ter iz pridobljenih regij ozadja izračuna histogram, s katerim se sestavi masko za filtriranje histograma oznake. Z sestavljeno masko filtriramo histogram oznake in s tem je postopek posodabljanja histograma zaključen. Na sliki 3.10 so predstavljeni izrezki regij ozadja, ki so označeni z modro barvo in izrezek oznake, ki je označen z zeleno barvo.

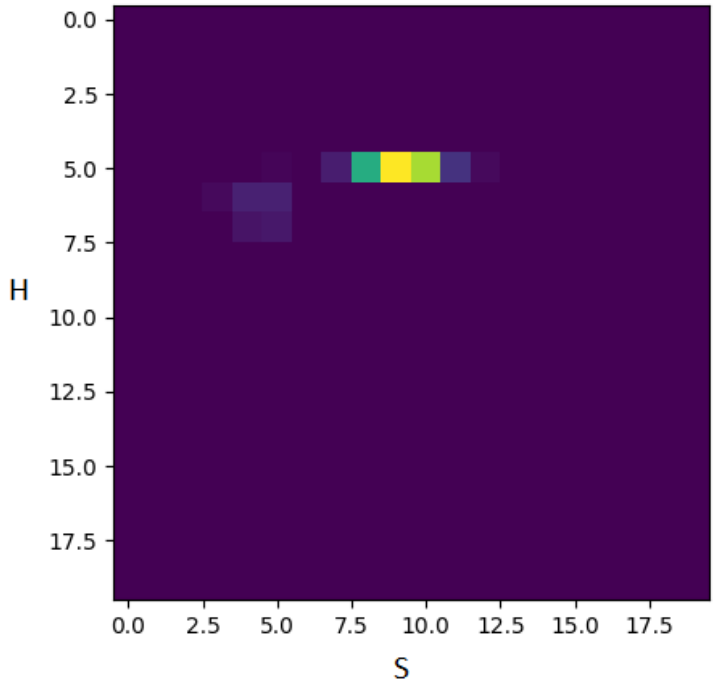


Slika 3.10: Zelena regija predstavlja oznako, modre regije predstavljajo regije ozadja.

3.4.7 Izračun histograma

Histogrami prikazujejo frekvenco pojavitve posamezne intenzitete barve na sliki. Barvni histogram se izračuna s preštevanjem intenzitet posameznega diskretiziranega barvnega kanala, ki se jih zbere v skupke določene velikosti (bins). Funkcija *calcHist*, ki je del knjižnice OpenCV, prejme kot vhod slike posameznih barvnih kanal, velikost posameznega skupka in razpon intenzitet posameznega barvnega kanala.

Za vsak kanal se prešteje število intenzitet, ki se jih razvrsti v pripadajoče skupke (bins). Rezultat funkcije je N dimenzionalen histogram, ki v vsakem intenzitetnem skupku hrani število pojavitev določene intenzitete. Dimenzija histograma je odvisna od števila barvnih kanalov, ki jih vsebuje slika. Iz sivinske slike dobimo 1 dimenzionalen histogram, medtem ko iz HSV slike dobimo 3 dimenzionalen histogram. Primer histograma, ki je sestavljen iz kanala barvni odtenek (hue) in intenzitete (saturation), je prikazan na sliki 3.11. Podroben opis računanja histogramov je predstavljen v [53].



Slika 3.11: 2D histogram v barvnem prostoru HSV: S je intenziteta, H predstavlja barvni odtenek.

Poglavje 4

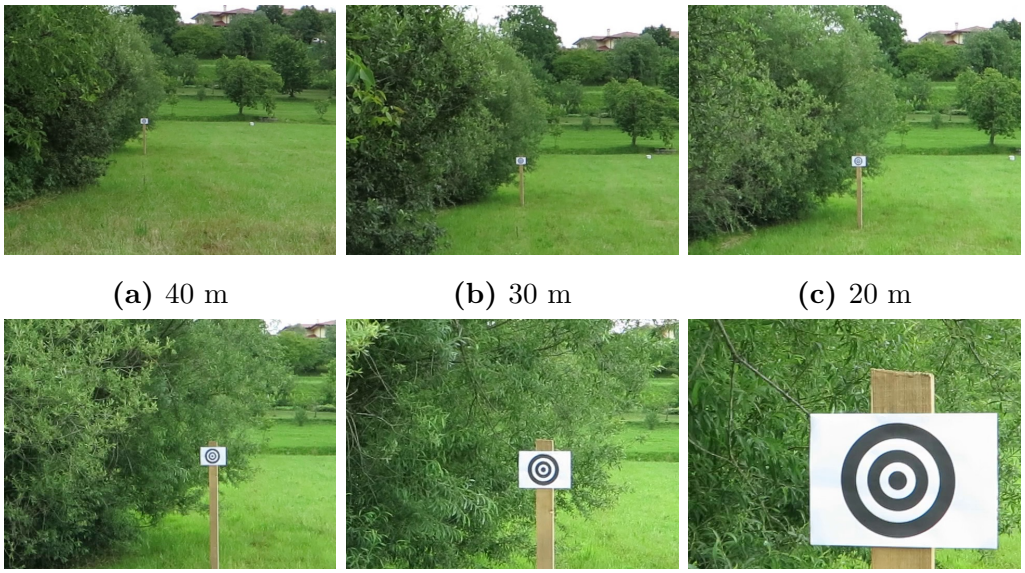
Rezultati

V tem poglavju predstavimo rezultate implementiranega sistema za detekcijo in sledenje oznaki. V prvem delu poglavja predstavimo testno okolje, s katerim smo ovrednotili naše delo. Rezultate razdelimo na več sklopov, ki se razlikujejo glede na vrsto testiranja. Najprej ovrednotimo natančnost, robustnost in časovno zahtevnost detekcijskega in sledilnega algoritma. Poleg tega ovrednotimo tudi delovanje celotnega sistema skupaj in predstavimo rezultate algoritma, ki je pokazal najboljše rezultate. Ovrednotenje delovanja celotnega sistema nadaljujemo z različnimi testnimi scenariji. Najprej ovrednotimo delovanje sistema pri uporabi dveh različno zasnovanih oznak. Nato sistem ovrednotimo z različnimi videoposnetki, ki so snemani iz določene razdalje in pod določenimi koti. Poleg tega opravimo umetno simulacijo različnih pogojev, kot so spremembe v osvetljenosti, zamegljenosti in dodajanju megle. Dodatno ovrednotimo sistem še z videoposnetki drugačnih ozadij.

4.1 Priprava okolja za ovrednotenje

Za potrebe ovrednotenja implementiranega sistema je bilo potrebno pripraviti okolje, ki predstavlja približek realnim razmeram. Pripravljeno okolje predstavljajo videoposnetki, ki simulirajo letenja letalnika. Videoposnetki so

posneti tako, kot če bi letalnik snemal let s pomočjo nameščene kamere. Primer sličic enega videoposnetka je prikazan na sliki 4.1. Pri tem je potrebno poudariti, da je tudi okolica pripravljena v manjšem merilu od realnega zaradi praktičnosti in omejenih virov.



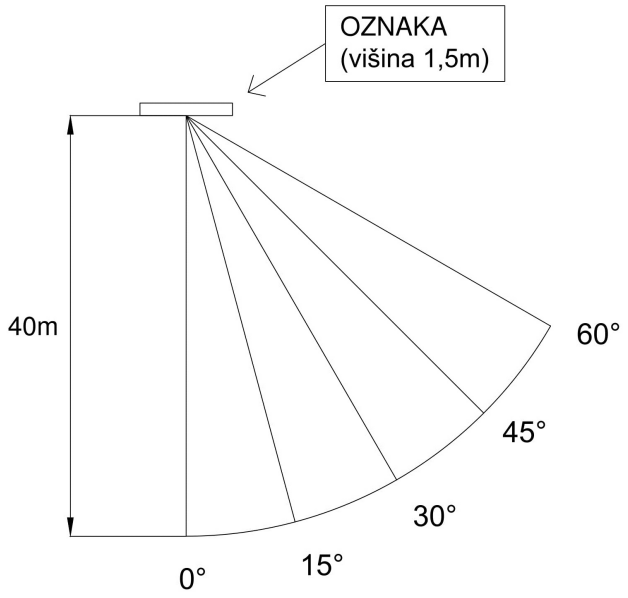
(a) 40 m **(b)** 30 m **(c)** 20 m

(d) 10 m **(e)** 5 m **(f)** 1 m

Slika 4.1: Sličice videoposnetka na različnih razdaljah.

Realno okolje zajema oznako velikosti do največ 5x8 metrov, ki bi bila snemana z razdalje najmanj 1 km. Posnetke bi bilo s pomočjo letalnika s fiksnimi krili težko zajeti, saj ima brezpilotni letalnik omejene manevrske sposobnosti in je cenovno predrag.

S simuliranimi pogoji v okolju želimo ovrednotiti delovanje celotnega sistema, ki je razdeljen na detekcijo in sledenje znane oznake. Snemanje videoposnetkov je potekalo tako, da smo zasnovano oznako velikosti A4 formata lista papirja postavili na travnik na višino 1,5 m od tal. Posneli smo 10 videoposnetkov oznake iz različnih kotov, in sicer po dva posnetka na posamezen kot. Območje smo razdelili na 5 kotov. Prvi kot 0 ° predstavlja snemanje videoposnetka pravokotno na oznako. Ostali videoposnetki so posneti na vsakih 15 ° v desno od oznake do kota 60 °, kot je prikazano na sliki 4.2.



Slika 4.2: Tloris postavitve snemalnega okolja.

Vsi videoposnetki so posneti iz začetne razdalje 40 metrov do razdalje 1 meter. V tabeli 4.1 so predstavljene razlike velikosti in oddaljenosti oznak ter razliko v hitrosti gibanja med simuliranim in realnim okoljem. Podatki v tabeli so približki izračunani na podlagi razmerja med velikostjo oznake v simuliranem ter realnem okolju. Razmerje med velikostjo oznake je približno 1:25.

Tabela 4.1: Mere v simuliranem in realnem okolju.

	Simulirano okolje	Realno okolje
Velikost oznaka	21,0x29,7cm	5x8m
Oddaljenost oznake	40,20,10m	1000,500,250m
Hitrost gibanja	0,65m/s	16,25m/s

Ker je snemanje potekalo na prostem, ni mogoče zagotoviti popolnoma enakih razmer na vsakem posnetku. Različne pogoje v naravi smo simulirali

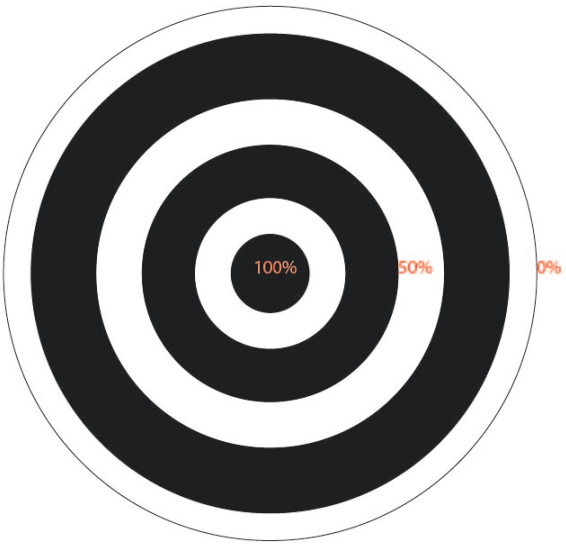
z umetnim dodajanjem ali odstranjevanjem informacij iz videoposnetkov in s tem omogočili natančno ovrednotenje implementiranega sistema v nadzorovanem okolju. Videoposnetke smo posneli s fotoaparatom Canon G16 v resoluciji 1920x1080 slikovnih elementov s 30 sličicami na sekundo. Vse videoposnetke smo pred izvedbo ovrednotenja zmanjšali z rezanjem posamezne sličice na velikost 640x480 slikovnih elementov. Z orodjem VOT annotator [57] smo ročno označili pozicijo in velikost oznake na vseh videoposnetkih, ki sta potrebni za ovrednotenje implementiranega sistema. Označevanje pozicije oznake je bilo izvedeno na razmaku 3 sličic. Za določitev pozicije oznake na ostalih neoznačenih sličicah je poskrbel sam VOT annotator s pomočjo interpolacije.

Ovrednotenje implementiranega sistema je potekalo na posnetih videoposnetkih. Sistem smo ovrednotili z relativno mero oddaljenosti med ročno označenim centrom oznake in centrom, ki ga vrne sistem. Oddaljenost med točko pravega centra in centra sistema smo izračunali z enačbo (4.1). Ker se velikost oznake skozi sličice videoposnetka spreminja, je bilo potrebno natančnost sistema oceniti z enačbo (4.2). Enačba natančnosti izračuna procentualno natančnost tako, da oddaljenost med centrom sistema in pravim centrom delimo s polovično višino oznake. Slika 4.3 (a) prikazuje mero natančnosti na oznaki, medtem ko slika 4.3 (b) prikazuje primer rezultatov sistema z določanjem centra oznake.

$$distance = \sqrt{(x1 - x2)^2 + (y1 - y2)^2} \tag{4.1}$$

$$accuracy = distance/(markerHeight/2) \tag{4.2}$$

Za ovrednotenje sistema smo pripravili en scenarij, ki oceni učinkovitost posameznih delov sistema. Drugi scenarij oceni delovanje sistema na dveh različnih oznakah. Nato ocenimo delovanje sistema pod različnimi koti. Četrti scenarij preveri natančnost delovanja z videoposnetkom, na katerem umetno odvzemamo in dodajamo osvetljenost. Pri petem scenariju ravno tako umetno deformiramo sliko s tem, da jo zameglimo. Za šesti scenarij ocenimo sistem z dodajanjem megle in pri zadnjem, sedmem scenariju



(a) Natančnosti na oznaki.

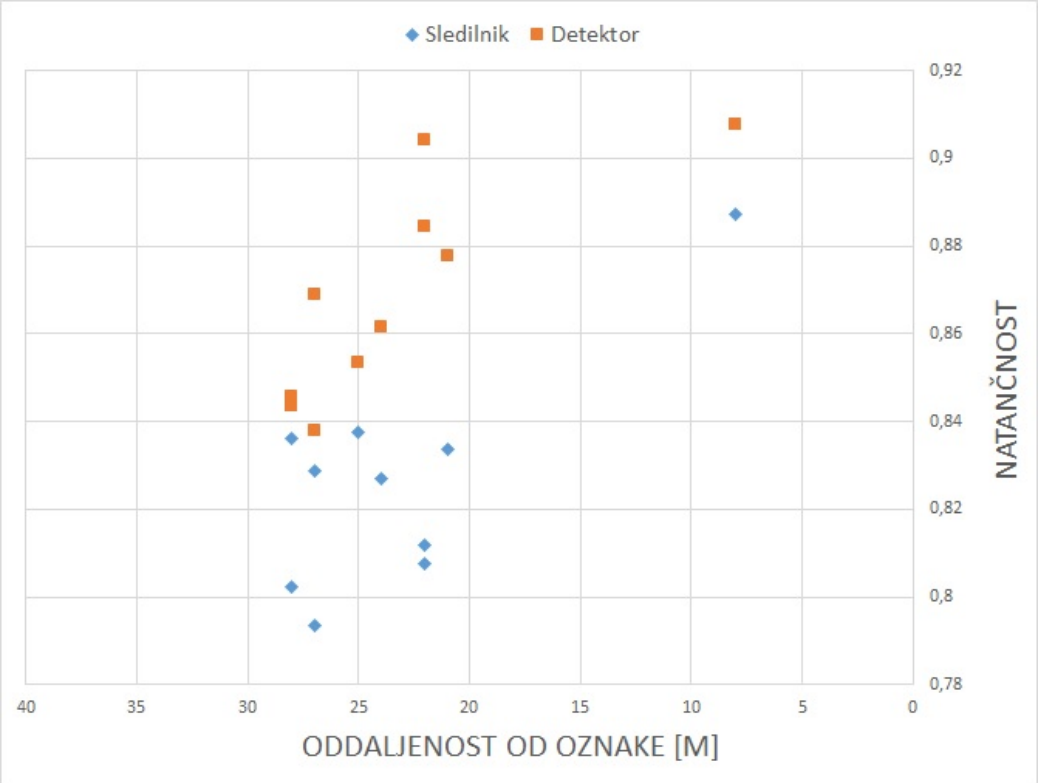
(b) Rezultati sistema z določanjem centra oznake.

Slika 4.3: Mera natančnosti in primer rezultatov sistema na oznaki.

ocenimo delovanje sistema, če zamenjamo ozadje oznake v videoposnetku z ozadjem drugih videoposnetkov.

4.2 Ovrednotenje posameznih delov sistema

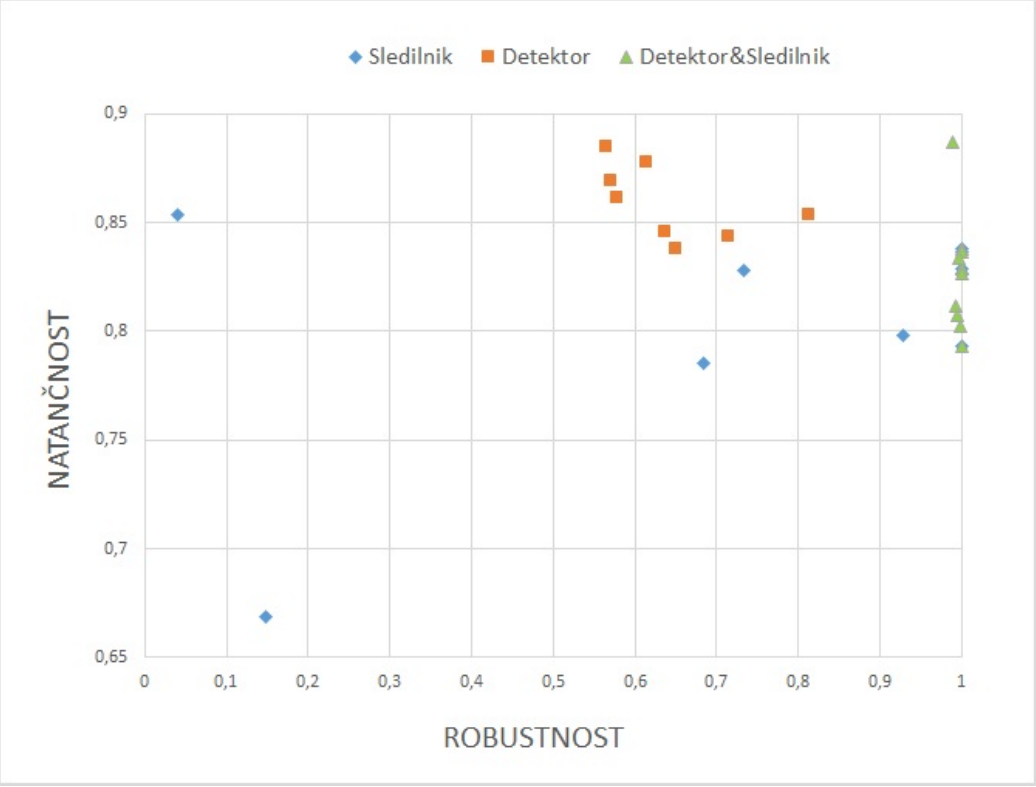
Ovrednotenje poteka tako, da se ločeno oceni delovanje detektorja, sledilnika in kombinacijo obojega kot celoto. Ovrednotenje posameznih delov sistema je potekalo na posnetih videoposnetkih. Z ovrednotenjem delovanja posameznih delov in sistema kot celote smo ovrednotili učinkovitost, natančnost in robustnost sistema ter na podlagi ocen izbrali del oziroma celoto, ki vrača najboljše rezultate. Najprej se je izvajanje sledilnika in detektorja preverilo na vseh posnetih videoposnetkih, ki vključujejo po dva posnetka za vsak kot.



Slika 4.4: Primerjava natančnosti med detektorjem in sledilnikom.

Iz rezultatov je bilo mogoče izračunati natančnost detektorja in sledilnika v odvisnosti od oddaljenosti od oznake. Rezultati, ki prikazujejo natančnost detektorja in sledilnika v odvisnoti od oddaljenosti od oznake, so prikazani na sliki 4.4. Vsaka pika na grafu prikazuje povprečno natančnost čez celoten videoposnetek od prve uspešne detekcije oznake naprej. Iz grafa je razvidno, da so natančnosti detektorja, ki ga predstavljajo oranžne pike, vedno boljše od natančnosti sledilnika. Razlike niso velike, ampak so očitne. Natančnosti pri detektorju so boljše od sledilnika, ker detektira elipse na oznaki, ki bolje določajo center oznake, kot pa sledilnik, ki predstavlja center oznake s središčem pravokotnika.

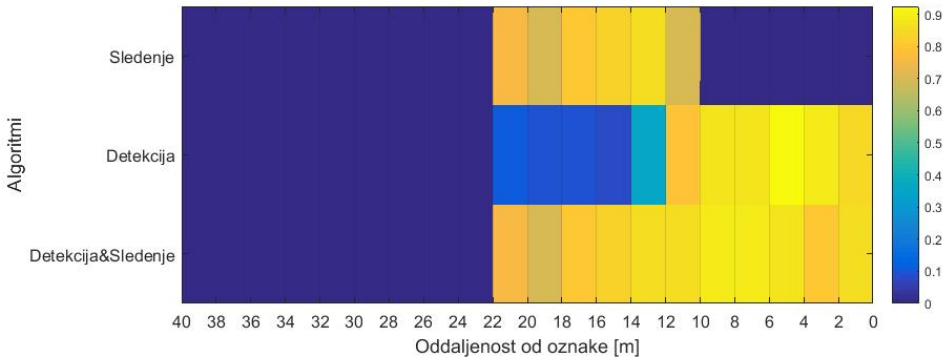
Na sliki 4.5 je prikazana natančnost detektorja, sledilnika in kombinacije obojega v odvisnoti od robustnosti.



Slika 4.5: Primerjava natančosti in robustnosti detektorja, sledilnika ter celotnega sistema.

Robustnost je mera, ki predstavlja relativno število uspešnih zaznav oznake na intervalu sličic videoposnetka od prve uspešne detekije do konca sličic videoposnetka. Ovredontenje robustnosti in natančnosti je bilo ravno tako izvedeno na vseh videoposnetkih. Iz grafa je jasno razvidno, da je natančnost samega detektorja nekoliko boljša od sledilnika in kombinacije obojega. Poleg tega vidimo, da je robustnost pri sledilniku in detektorju slabša kot pri uporabi obojega skupaj. Robustnost pri sledilniku je slabša, ker ko sledilnik odpove, ga brez detektorja ni mogoče reinicializirati. Robustnost sledilnika dosega maksimalne vrednosti v primerih, ko sledilnik nikoli ne odpove. Pri detektorju je robustnost brez sledilnika slabša, ker pri večjih oddaljenostih od oznake detektor zazna oznako le vsakih nekaj sličic (cca. 20). S približevanjem oznaki se detekcija izboljša do vrednosti, ko detektor detektira oznako v vsaki zaporedni sličici videoposnetka. Najboljšo robustnost predstavlja kombinacija detektorja in sledilnika, ker detektor detektira oznako in nato prepusti delo sledilniku. Če sledilnik odpove, ga takoj reinicializira detektor in pri tem se v povprečju izgubi le 1 sličica, kar se odraža na grafu s robustnostjo nad 99%.

Za boljšo predstavo delovanja detektorja, sledilnika in kombinacije obeh algoritmov je na sliki 4.6 predstavljeno delovanje algoritmov na enem videoposnetku.



Slika 4.6: Rezultati testiranja delovanja detektorja, sledilnika in celotnega sistema na videoposnetku.

Slika predstavlja natančnost delovanja vseh treh kombinacij algoritmov v odvisnosti od oddaljenosti od oznake. Vsi trije začnejo delovati pri enaki oddaljenosti, ker za to poskrbi detektor v vseh treh primerih. Pri sledilniku je opaziti, da se zaradi zgrešitve sledenja delovanje sledilnika brez ponovne detekcije z uporabo detektorja zaključi. Pri detektorju je zaradi začetnih redkih detekcij oznake v posamezni sličici natančnost zelo nizka, ampak narašča s približevanjem oznaki. Detekcija pri krajših oddaljenostih deluje boljše, ker je oznaka bolje razvidna in jasna ter jo je enostavneje detektirati. Sistem, ki vsebuje detektor in sledilnik, se zaradi kombiniranja detekcije in sledenja odraža z dobrimi rezultati tudi na lokacijah, na katerih sta detektor in sledilnik sama po sebi neuspešna.

Slika 4.7: Povprečni čas izvajanja algoritmov sistema in standardna deviacija.

Poleg ovrednotenja natančnosti in robustnosti delovanja implementiranega sistema je pomembna tudi časovna zahtevnost. Sistem mora delovati v realnem času, da je mogoče s pomočjo sistema tudi krmiliti in voditi letalnik v oznako. Na sliki 4.7 so prikazane časovne ocene izvajanja detekcije

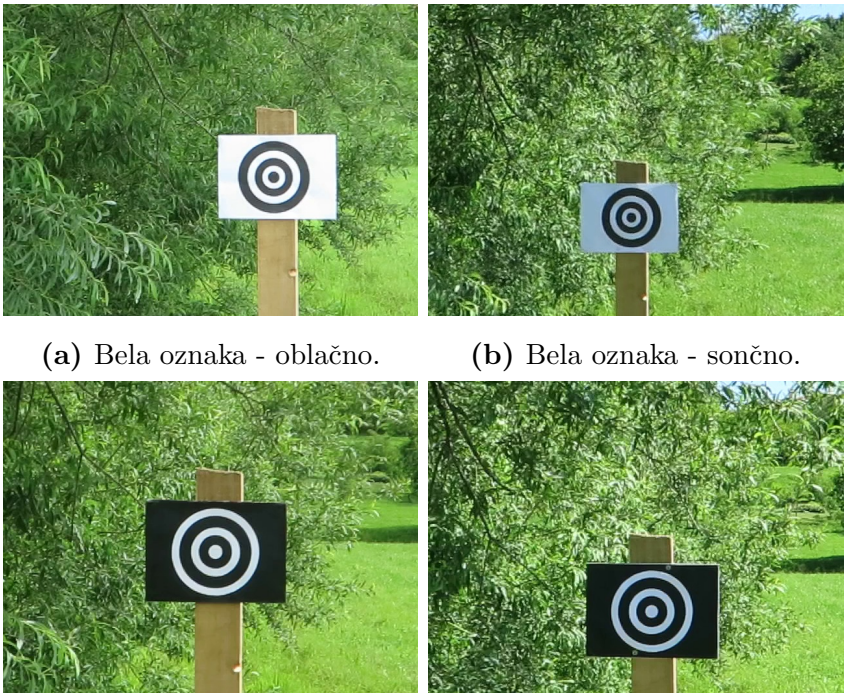
in sledenja na eni sličici videoposnetka. Meritve so bile izvedene na vseh desetih posnetih videoposnetkih in nato izračunana standardna deviacija ter povprečna vrednost izvajanja.

Izvajanje posameznega algoritma je ocenjeno v ms/sličico. Iz grafa na sliki 4.7 je razvidno, da je izvajanje sledilnika približno 20-krat hitrejše od izvajanja detektorja. Sledilnikovo izvajanje je veliko hitrejše zaradi enostavnejših operacij za sledenje ozanke in manjši sliki, ki jo mora sprocesirati. Kombinacija detektorja in sledilnika se v najboljšem primeru izvaja s povprečnim časom sledilnika, saj je v primeru takojšne detekcije oznake in mogočih le nekaj ponovnih redetekcijah, čas detekcije zanemarljiv. V najslabšem primeru bi moral sledilnik ves čas vračati nezanesljive rezultate, kar bi privedlo do povprečne hitrosti izvajanja, kot pri detektorju. Če vzamemo, da celoten sistem deluje približno polovico časa s hitrostjo detektorja in polovico časa s hitrostjo sledilnika, pomeni, da je hitrost celotnega sistema približno 9 ms/sličico, kar predstavlja izvajanje 111 sličic/s. Ta rezultat potrjuje zanesljivo delovanje sistema v realnem času.

Iz pridobljenih rezultatov detektorja, sledilnika in kombinacije obojega je razvidno, da je za natančno, učinkovito in hitro delovanje najprimernejša uporaba detektorja in sledilnika skupaj. V naslednjih simulacijah je ovrednotenje potekalo le na celotnem sistemu, ki ga sestavljata detekcijski in sledilni algoritem.

4.3 Ovrednotenje sistema na dveh oznakah

Sistem smo ovrednotili na dveh različnih oznakah in s tem preizkusili uspešnost ter natančnost delovanja sistema v odvisnoti od barv oznake. Obe oznaki sta krožni, ampak se razlikujeta v barvi ozadja in krožnic. Na sliki 4.8 sta prikazani obe oznaki posneti v sončnem in oblačnem vremenu.



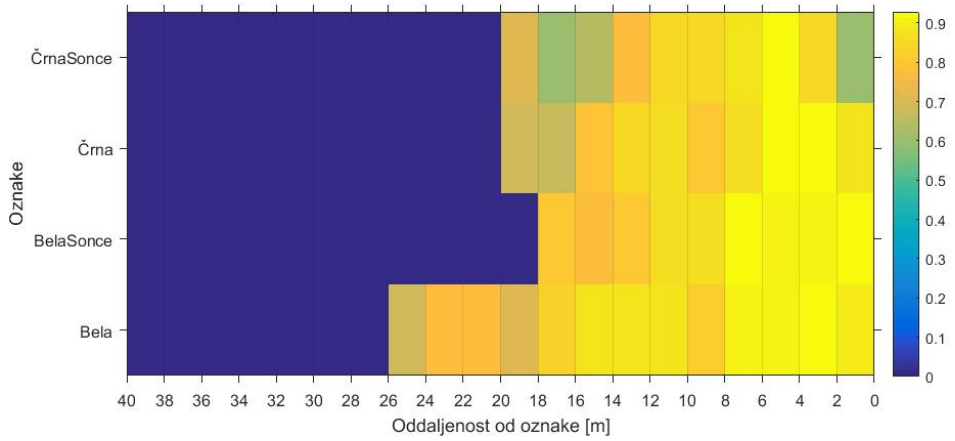
(a) Bela oznaka - oblačno. **(b)** Bela oznaka - sončno.

(c) Črna oznaka - oblačno. **(d)** Črna oznaka - sončno.

Slika 4.8: Belo-črna in črno-bela oznaka pri sončnem ter oblačnem vremenu.

Prva oznaka ima belo ozadje z črnimi krožnicami, medtem ko so pri drugi barve uporabljene ravno obratno. Druga razlika med obema oznakama je debelina krožnic. Pri oznaki z belim ozadjem so debelejše črne krožnice, kar zmanjša jakost pojava cvetenja. Enak pristop je uporabljen tudi pri oznaki s črnim ozadjem, pri kateri so bele krožnice tanjše.

Poleg barvnih odvisnosti oznake, je v ovrednotenje vključena še ocena natančnosti delovanja sistema v sončnem in oblačnem vremenu na obeh oznakah. Rezultati so prikazani na sliki 4.9. Os x predstavlja oddaljenosti od oznake, kar nam pove, na kateri oddaljenosti je začel sistem delovati. Vsaka vrstica na osi y predstavlja en videoposnetek, na katerem se je ovrednotilo delovanje sistema. Spodnji dve vrstici predstavljata delovanje sistema pri uporabi oznake z belim ozadjem in črnimi krožnicami, medtem ko zgornji dve vrstici predstavljata delovanje sistema pri uporabi oznake s črnim ozadjem in belimi krožnicami.



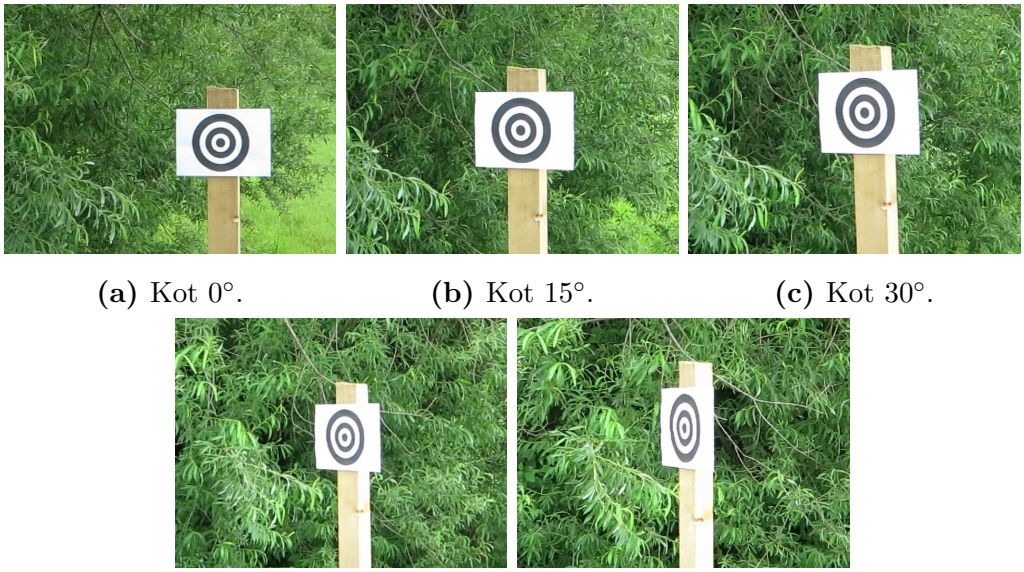
Slika 4.9: Rezultati ovrednotenja delovanja sistema z dvema različnima oznakama.

Iz grafa je razvidno, da sistem deluje najbolje pri oblačnem vremenu, ker sistem detektira oznako že pri 26 m. Delovanje sistema se občutno poslabša pri sončnem vremenu, saj se prva uspešna detekcija oznake izvede na 18 m. Pri oznaki s črnim ozadjem se glede na vremenske razmere delovanje sistema ne poslabša, ampak zaradi črne barve detektor detektira oznako na oddaljenosti 20 m. Iz rezultatov je razvidno, da se sistem najbolje odreže pri manjši izpostavljenosti oznake soncu. Natančnost delovanja sistema je boljša pri oznaki z belim ozadjem, vendar je konsistentnost boljša pri oznaki s črnim ozadjem. Oznaka s črnim ozadjem ima boljšo konsistentost, ker sence in različne osvetlitve oznake ne pridejo do tako velikega izraza, kot pri oznaki z belim ozadjem.

4.4 Ovrednotenje sistema pod različnimi koti

Delovanja sistema pod različnimi koti smo ovrednotili s posnetimi videoposnetki. Ovrednotenje je potekalo pod koti 0, 15, 30, 45 in 60 stopinj. Za vsak kot smo pridobili dvoje rezultatov iz dveh videoposnetkov snemanih pod enakim kotom. Videoposnetki posneti pod različnimi koti simulirajo le-

tenje letalnika proti oznaki pod različnimi koti. Sličice videoposnetkov, ki predstavljajo različne kote snemanja, so prikazane na sliki 4.10.

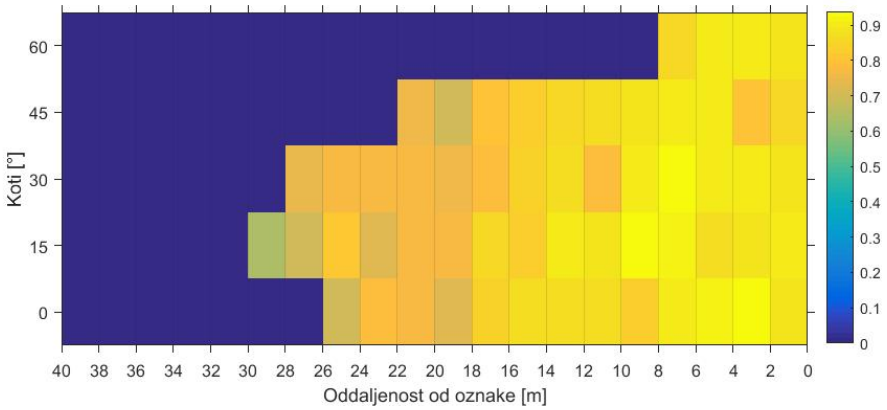


(a) Kot 0°. **(b)** Kot 15°. **(c)** Kot 30°.

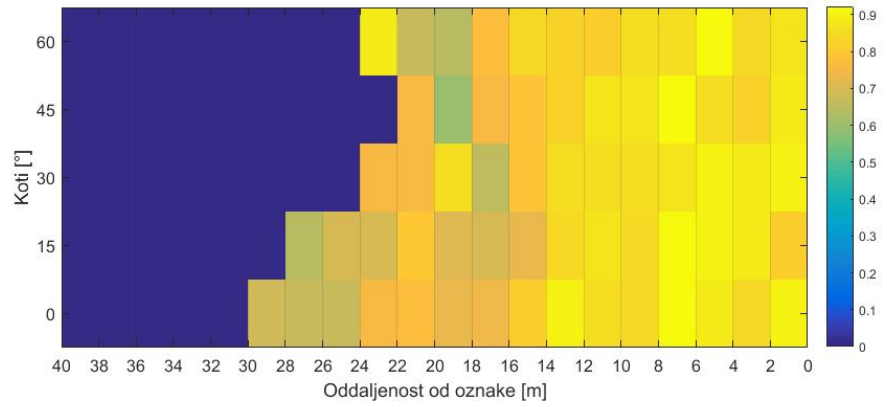
(d) Kot 45°. **(e)** Kot 60°.

Slika 4.10: Sličice videoposnetkov pri različnih kotih.

Rezultati ovrednotenja so prikazani na sliki 4.11. Na x osi je predstavljena oddaljenost od oznake v metrih. Vsaka vrstica na osi y predstavlja en videoposnetek, sneman pod kotom navedenim na osi. Na desni strani so napisane natančnosti zadetka sredine oznake od 0 % do 100 %. Grafa (a) in (b) prikazujeta rezultate izvedene na obeh množicah videoposnetkov. Iz grafov je razvidno, da se natančnost sistema povečuje s približevanjem oznaki in, da se razdalja pri kateri začne delovati sistem krajša s povečevanjem kota. Zanimiva sta videoposnetka pod kotom 0° in 60° na grafu (a). Sistem deluje na videoposnetku pod kotom 0° slabše kot na videposnetih pod kotom 15° ali 30°. Pri posnetku pod kotom 60° je opazno drastično slabše delovanje sistema, saj se iz dolžine 22 m začetka delovanja sistema pri kotu 45° zmanjša na 8 m pri kotu 60°. Slabše delovanje sistema pripisujemo nekoliko drugačnim razmeram v osvetljenosti in sencah na videoposnetku zaradi snemanja v naravi. Vidimo tudi, da se na grafu (b) natančnosti in razdalje v odvisnoti



(a) Skupina posnetkov 1.



(b) Skupina posnetkov 2.

Slika 4.11: Rezultati ovrednotenja delovanja sistema pod različnimi koti.

od kotov sorazmerno povečujejo oziroma krajšajo, kar je pričakovano. Poleg tega je razvidno, da je delovanje robustno do kota 45°.

4.5 Ovrednotenje sistema različnih osvetljenosti

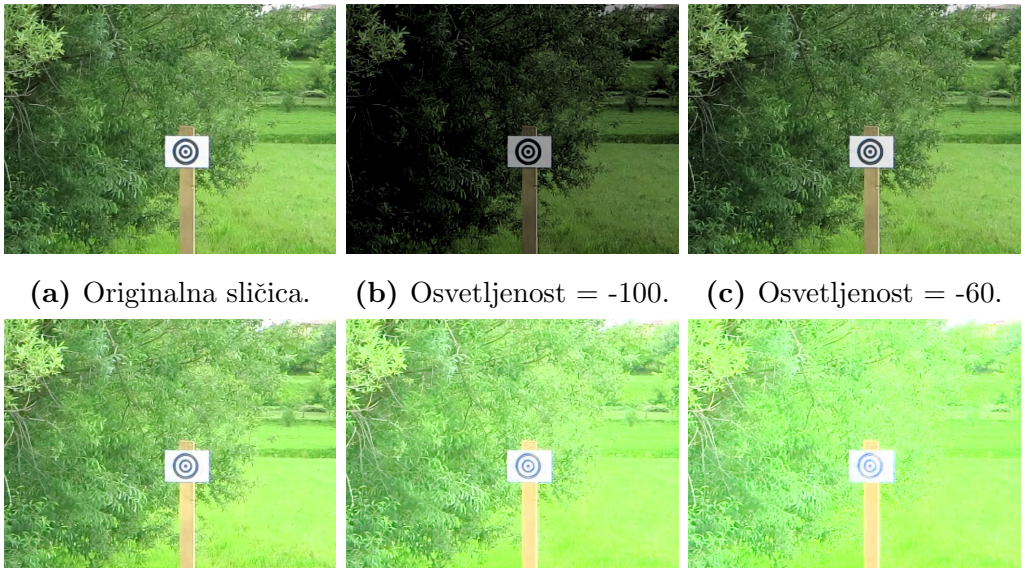
Za ovrednotenje delovanja sistema pri različnih osvetljenostih sličic smo vzeli dva videoposnetka iz množice posnetih videoposnetkov. Oba videoposnetka sta snemana pravokotno na oznako. Simuliranje različne svetlobe v videoposnetkih smo izvedli s pretvorbo sličic videoposnetka v HSV barvni prostor. Na vsaki sličici videoposnetka smo prišteli kanalu V (Value) vrednost Illum, kot je prikazano v enačbi 4.3 ter sličico pretvorili nazaj v RGB barvni prostor.

$$I(i,j,2) = \begin{cases} I(i,j,2) + Illum & \text{if } 255 - I(i,j,2) > Illum \\ 255 & \text{otherwise} \end{cases} \tag{4.3}$$

Primer spremembe osvetljenosti v sličici videoposnetka je prikazan na sliki 4.12.

Rezultati delovanja sistema pri spreminjanju osvetljenosti na dveh videoposnetkih so predstavljeni na sliki 4.13. Grafa (a) in (b) prikazujeta rezultate posameznih videoposnetkov. Na osi y so predstavljene vrednosti spremembe osvetlitve na posamezni sličici. 0 predstavlja originalen videposnetek, medtem ko je maksimalna sprememba osvetljenosti 140. Od vrednosti 140 naprej sistem ne več deluje. Za potemnitev smo uporabili vrednosti do -100. Os x predstavlja razdalje, na kateri začne delovati sistem in barvna lestvica na desni predstavlja natančnost sistema na posameznih oddaljenostih od oznake.

Pri potemnjenih sličicah videoposnetka so oddaljenosti, pri kateri sistem prvič detektira oznako ne bistveno razlikujejo, ker je oznaka črno-bela. Črnobela oznaka ostane vidna zaradi visoke kontrastnosti, medtem ko se okolica potemnjuje in z višjimi vrednostmi potemnitve postane črna. Boljše rezultate detekcije oznake pri potemnitvah sličic videoposnetka ni mogoče doseči, ker s tem ne izboljšamo jasnosti slike ter so zaradi tega linije elips oznake na



(a) Originalna sličica. **(b)** Osvetljenost = -100. **(c)** Osvetljenost = -60.

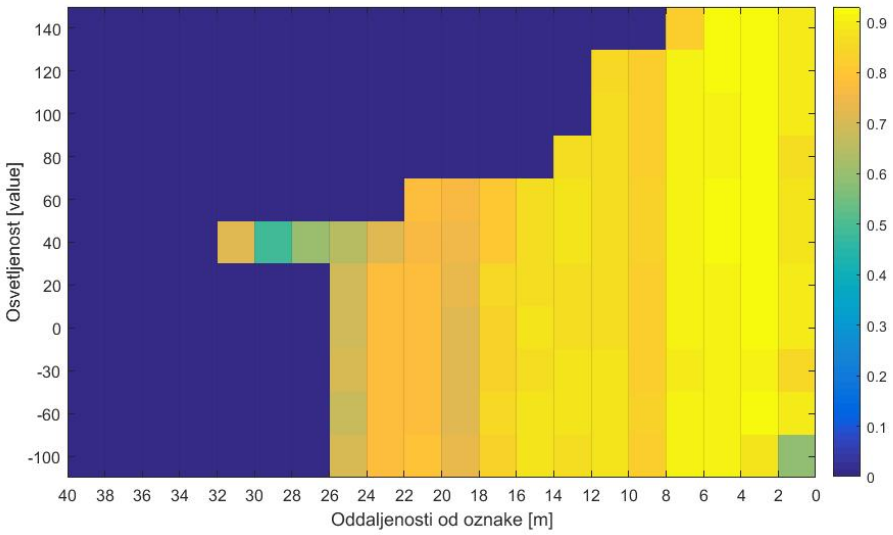
(d) Osvetljenost = 60. **(e)** Osvetljenost = 100. **(f)** Osvetljenost = 140.

Slika 4.12: Sprememba osvetljenosti v sličicih videoposnetka za simuliranje svetlobe.

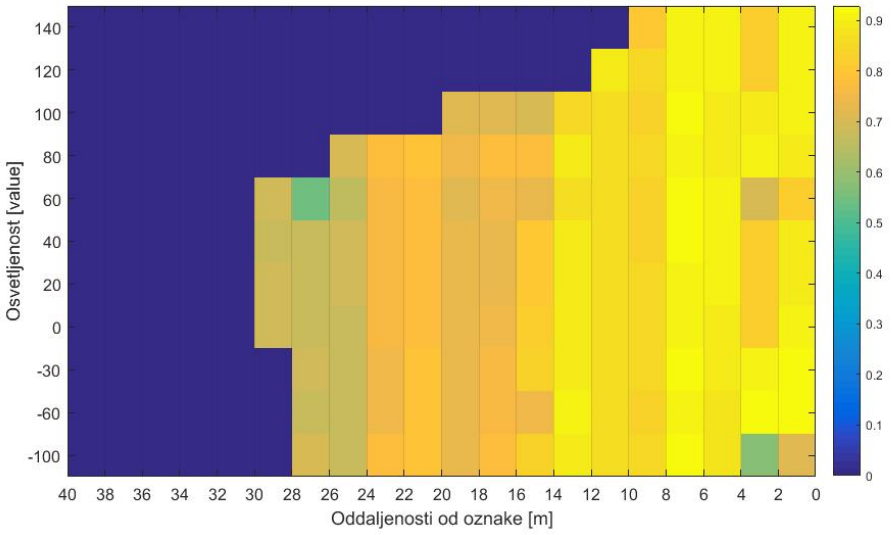
daljših razdaljah še vedno nejasne. Pri vrednosti osvetljenosti -100 se na obeh grafih vidi padec natančnosti od 4 m oddaljenosti od oznake do 1 m. Do tega pride zaradi senc na oznaki, ki povzročijo, da nastanejo črne regije na oznaki. Črne regije na oznaki prekinejo linije elips, kar povzroči slabše prepoznavo oznake in posledično slabšo natančnost sledilnika.

Natančnost sistema se povečuje s približevanjem oznaki in razdalja, pri kateri začne delovati sistem se krajša s povečevanjem osvetljenosti na sličicah videoposnetka. Na grafu (a) je rezultat sistema pri vrednosti osvetljenosti 40 iztopajoč. Do tega je prišlo, ker je zaradi spremembe osvetljenosti, oznaka bolje opazna in jo je sistem prej detektiral. To še ne pomeni, da bi sistem bolje deloval z umetnim povečevanjem osvetljenosti v sličicah videoposnetkov, ker se v povprečju delovanje sistema slabša.

Na grafu (b) je razvidno, da se do vrednosti osvetljenosti 60 delovanje sistema ne bistveno poslabša. Od vrednosti 60 naprej pa se začetna točka delovanja sistema bistveno skrajša iz ene stopnje osvetljenosti v drugo. Neka-



(a) Posnetek 1.



(b) Posnetek 2.

Slika 4.13: Rezultati ovrednotenja delovanja sistema pri spreminjaju osvetljenosti v videoposnetkih.

tere osamele natančnosti sistema, ki iztopajo s temnejšo barvo predstavljajo, da je v originalnem posnetku na tisti oddaljenosti prišlo do odboja svetlobe ali sence na oznaki, kar je vplivalo na natančnost sistema.

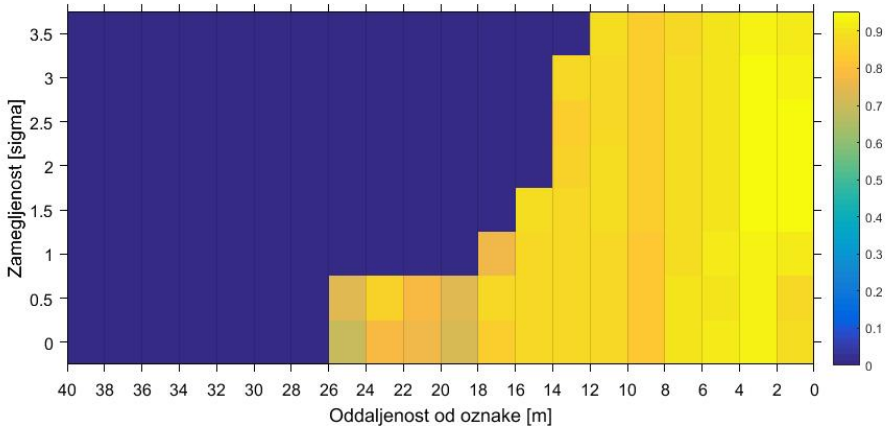
4.6 Ovrednotenje sistema pri različni zamegljitvi

Za simulacijo hitrih gibov in tresljajev kamere smo uporabili Gaussov filter, s katerim smo sličice videoposnetka zameglili (blur). Ovrednotenje delovanja sistema z zamegljitvijo smo opravili na dveh videoposnetkih, snemanimi pravokotno na oznako. Zamegljenost smo opravili z Gaussovim filtrom z velikostjo jedra 5x5, pri katerem smo spreminljali vrednost σ. Slika 4.14 prikazuje primer zamegljenosti sličice videoposnetka.

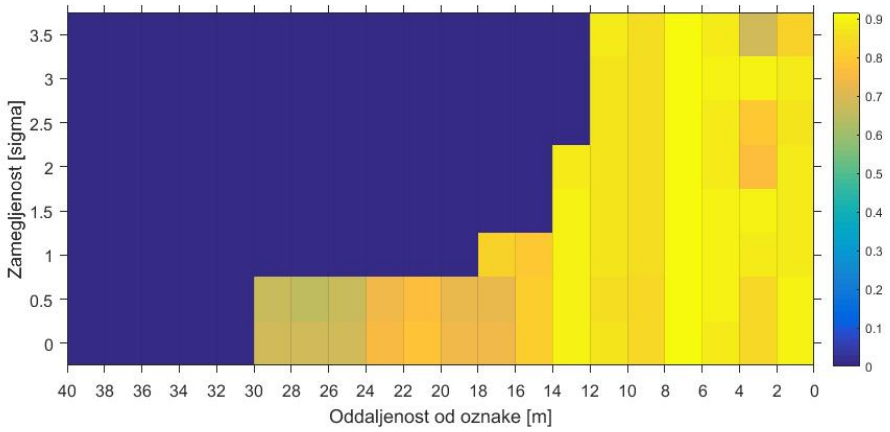
(a) Originalna sličica videoposnetka **(b)** Faktor $\sigma = 1,5$ **(c)** Faktor $\sigma = 3,5$

Slika 4.14: Pogladitev sličice videoposnetka za simuliranje zamegljenosti.

Rezultati delovanja sistema so prikazani na sliki 4.15. Grafa (a) in (b) prikazujeta rezultate, ki so bili opravljeni na dveh videoposnetkih. Na osi y so prikazani faktorji σ ter na osi x oddaljenosti od oznake merjeno v metrih. Z barvo je predstavljena natančnost delovanja sistema na posameznih razdaljah od oznake pri zamegljitvah z določeno vrednostjo σ. Na obeh grafih je razvidno, da se povečevanjem zamegljenosti s faktorjem σ krajša začetna točka delovanja sistema. Od faktorja σ 3,5 naprej se delovanje ne poslabša, ampak sistem detektira oznako pri enaki oddaljenosti kot pri faktorju σ 3,5.



(a) Posnetek 1.



(b) Posnetek 2.

Slika 4.15: Rezultati ovrednotenja delovanja sistema pri različnih zameglje-nostih videoposnetka.

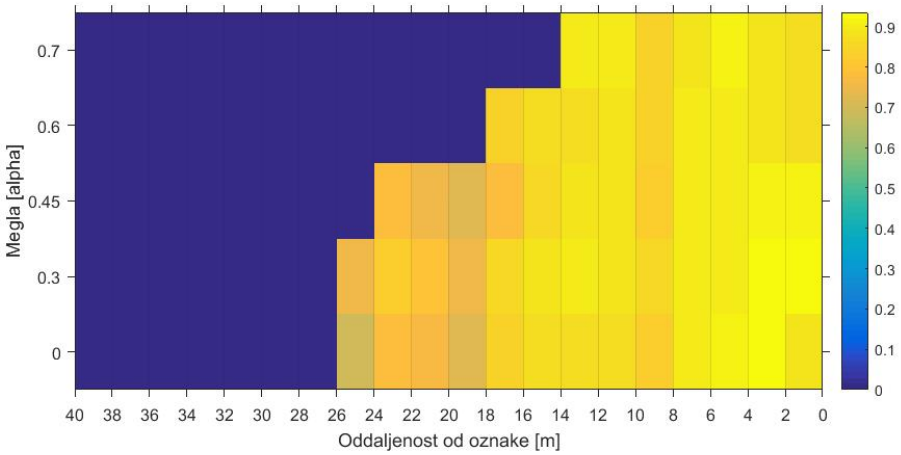
4.7 Ovrednotenje sistema z dodajanjem megle

Ovrednotnje delovanja sistema smo izvedli tudi na videoposnetkih z dodano meglo. Za simulacijo megle smo na belo sliko naključne slikovne elemente pobarvali z nizkimi intenzitetami (skoraj črni slikovni elementi) in sliko pogladili z Gaussovim filtrom. Generirana slika je predstavljala meglo, ki smo jo nato z različnimi faktorji transparentnosti dodali na originalne sličice videoposnetka. Sistem smo ovrednotili na dveh videoposnetkih z dodano meglo. Slika 4.16 prikazuje simulacijo megle z različnimi faktorji α na sličicah videoposnetka.

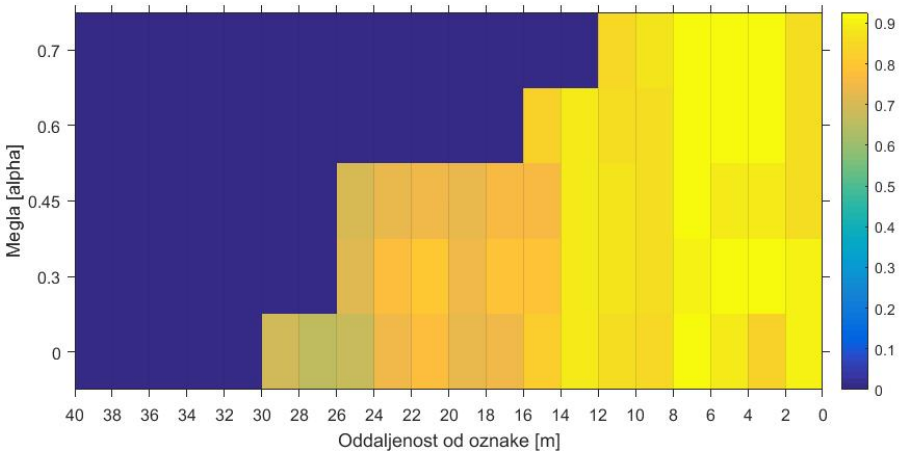
(a) Original **(b)** Faktor $\alpha = 0.3$ **(c)** Faktor $\alpha = 0.6$ **(d)** Faktor $\alpha = 0.7$

Slika 4.16: Simuliranje megle na sličici videoposnetka.

Rezultati delovanja sistema so prikazani na sliki 4.17. Grafa, podobno kot pri prejšnjih grafih, prikazujeta natančnost sistema v odvisnoti od faktorja transparentnosti in oddaljenosti od oznake. Faktor transparentnosti α pri vrednosti 0 predstavlja originalno sliko, medtem ko faktor α 0,7 originalno sliko prekrije z umetno sliko megle v razmerju 30 % originalna slika ter 70 % slika megle. Na obeh grafih je razvidno, da se s povečevanjem megle krajša razdalja, pri kateri začne delovati sistem. Pri faktorju α višjem od 0,7 sistem ne deluje več. Kot pri ostalih grafih, se tudi tukaj natančnost sistema povečuje s približevanjem oznaki.



(a) Posnetek 1.



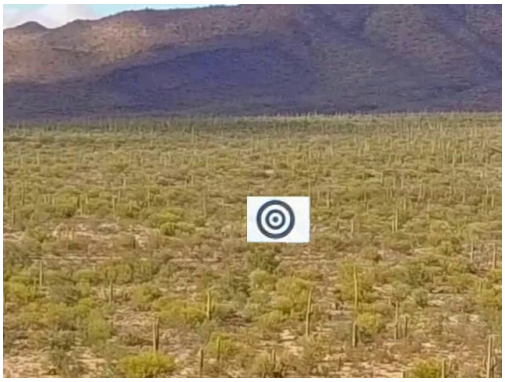
(b) Posnetek 2.

Slika 4.17: Rezultati ovrednotenja delovanja sistema z umetnim dodajanjem megle v videoposnetka.

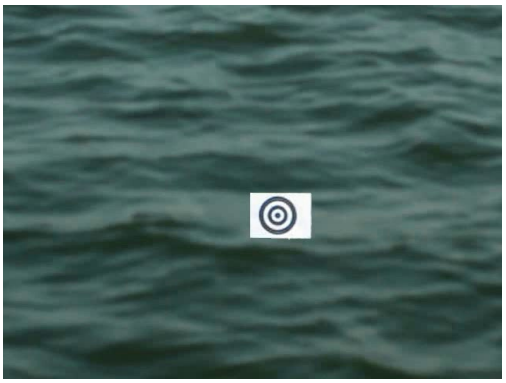
4.8 Ovrednotenje sistema na različnih posnetkih

Delovanja sistema smo do sedaj ovrednotili le na videoposnetkih, ki so bili posneti na travniku. Da bi ocenili delovanje sistema v različnih okoljih, je bilo potrebno pridobiti videoposnetke z drugačno okolico okrog oznake. Najprej je bilo potrebno pridobiti različne videoposnetke iz narave. Videoposnetki so bili pridobljeni iz zbirke brezplačnih videoposnetkov na spletnih straneh [58] in [59]. Iz zbirke videoposnetkov smo izbrali videoposnetke, ki so posneti v okoljih različnih barv. Izbrani videoposnetki prikazujejo gozd, puščavo in vodo. Sličice izbranih videoposnetkov so prikazane na sliki 4.18.

(a) Gozd [58].



(b) Puščava [58].

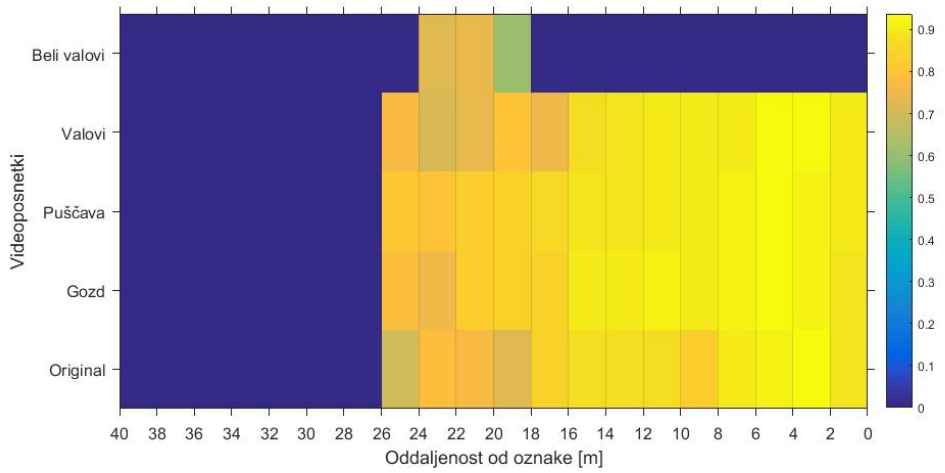


(c) Valovi [58].

(d) Beli valovi [59].

Slika 4.18: Sličice dodatnih videoposnetkov.

V vsak izbran videoposnetek je bilo potrebno dodati zasnovano oznako za ovrednotenje delovanje sistema. Iz naše zbirke posnetih videoposnetkov smo izbrali videoposnetek, ki je bil sneman pri kotu 0° in iz vsake sličice videoposnetka pridobili le oznako in ozadje zavrgli. Za pridobivanje oznake je bila uporabljena funkcija *GrabCut* iz knjižnice OpenCV. Natančno delovanje funkcije *GrabCut* je predstavljeno v [34]. Vsako pridobljeno oznako



Slika 4.19: Rezultati ovrednotenja delovanja sistema na videoposnetkih z različnimi ozadji.

iz posamezne sličice videoposnetka smo prilepili na sličice novo izbranih videoposnetkov. Pozicija oznake v novih videoposnetkih je zaradi tega ostala enaka kot pri originalnem posnetku.

Rezultati ovrednotenja delovanja sistema na pripravljenih videoposnetkih so prikazani na sliki 4.19. Graf prikazuje natančnost sistema glede na oddaljenost od oznake na osi x in glede na različne videoposnetke na osi y. Iz grafa je razvidno, da ne glede na videoposnetek, sistem začne delovati pri enaki razdalji z izjemo videoposnetka Whitewaves. Pri videoposnetku Whitewaves je sistem od oddaljenosti 18 m navzdol odpovedal. Do odpovedi je prišlo zaradi prevelike podobnosti barv ozadja in oznake. Ker so valovi na vrhu bele barve, je sledilnik zajel še ozadje. Pogoji za ponovno detekcijo niso bili

izpolnjeni, zato se je sledenje še naprej izvajalo, kar je povzročilo napačno delovanje celotnega sistema.

Za določitev zanesljivosti sledilnika bi bilo potrebno narediti še veliko raziskav in bolje opredeliti pogoje, ki bi določali zanesljivo sledenje. Ena od rešitev je tudi, da bi sistem deloval le z detektorjem, kar bi poslabšalo robustnost sistema, ampak bi kljub težkim razmeram za sledenje sistem deloval. Druga možna rešitev je izboljšati sledilnik z dodatnimi sledilnimi metodami ali z zamenjavo oznake, ki bi bila bolj razločna od ozadja. Za dodatno sledilno metodo bi lahko uporabili optični tok, ki bi poleg barve spremljal še gibanje oznake v sliki.

Poglavje 5

Zaključek

5.1 Sklepne ugotovitve

Brezpilotni letalniki s fiksnimi krili so se v zadnjih desetih letih razširili iz vojaškega področja na druga področja, ki so na voljo vsem ljudem. To so predvsem gospodarske in komercialne panoge, pri katerih se tovrstni letalniki uporabljajo za različna opravila. Cilj našega dela je bil razviti sistem, ki bo s pomočjo računalniška vida pripomogel k realizaciji avtonomnega pristajanja brezpilotnih letalnikov s fiksnimi krili. Trenutne sposobnosti nekaterih letalnikov so omejene na pristajanje s padalom, kar onemogoča pristajanje na premikajočih se platformah in lokacijah z majhno pristajalno površino. Za odpravo teh težav smo si zastavili cilj razviti sistem, ki bo omogočal letalniku pristanek v mrežo. Sistem sestavljata detektor in sledilnik oznake, ki bo z informacijami o poziciji oznake na sliki omogočal krmiljenje letalnika.

Razvoj sistema se je začel pri zasnovi oznake za potrebe detekcije in sledenja. Zasnova oznake je ključnega pomena za kakovost delovanja sistema. Pri zasnovi oznake so se izkazale za zelo pomembne njene lastnosti. Lastnosti, kot so oblika, barva in velikost oznake imajo velik vpliv na delovanje sistema v različnih pogojih. Oznaka s kocentrični krožnimi vzorci je enostavna za prepoznavo in primerna za detekcijo in sledenje v naravi zaradi distinktivnosti od okolice. Za barvo se je izkazala najbolje oznaka z be-

lim ozadjem in črnimi krožnicami, pri kateri je potrebno poudariti, da je bela barva podvržena bleščanju in efektu cvetenja. Rešitev tega problema je bila v uporabi debelejših črnih krožnic, ki so bolje vpijale svetlobo in tako zmanjšale odboje.

Rešitev za detekcijo krožne oznake je v iskanju robov in obrisov v sliki, preko katerih je bilo mogoče izluščiti elipse, ki so predstavljale zasnovano oznako. Z razvitim detekcijskim algoritmom smo dosegli uspešno detekcijo oznake pod različnimi koti na oddaljenosti približno 25 m od oznake, kar v realnem okolju predstavlja detekcijo na oddaljenosti okrog 600 m. Opaziti je občuten padec uspešnosti detekcije pri kotu 60°, kar je povsem normalno pri tako velikem faktorju popačenosti oznake. Detekcija se je dobro odrezala tudi pri ostalih simuliranih pogojih, kot so megla, zamegljenost in sprememba osvetljenosti. V vseh primerih se je pri višjih stopnjah popačenja originalne slike uspešnost in natančnost detekcije zmanjšala. Kljub težjim pogojem zaradi popačenja je detekcijski algoritem dosegel raven pričakovanj.

Detekcijski algoritem deluje nad celotno sliko, kar posledično vpliva na čas izvajanja algoritma. Za pohitritev sistema smo implementirali sledilni algoritem, ki deluje na principu iskanja lokalnega maksimuma v projekcijski sliki. Zaradi spreminjajoče velikosti oznake v videoposnetku smo uporabili algoritem CamShift, ki adaptivno prilagaja velikost okna za označitev oznake. Implementirani sledilnik je opravljal sledenje tudi do 20-krat hitreje od detektorja in pri tem je natančnost sistema padla za zanemarljiv faktor. Sledilnikova slabost je odpoved delovanja v primeru, da izgubi oznako. Zaradi tega je bilo potrebno zasnovati metodo za določitev zanesljivosti sledenja in v primeru nezanesljivega rezultata ponovno aktivirati detektor, da se je lahko sledilnik ponovno inicializiral. Sledilnik je zelo hiter in deluje dobro pod različimi pogoji, vendar je podvržen napačnemu sledenju pri ozadjih, ki so podobne barve od sledene oznake.

5.2 Nadaljnje delo

Trenutne težave sistema se nahajajo predvsem v sledilnem algoritmu. Sledilnik odpove, če sta barva oznake in barva ozadja enakih odtenkov. Enostavna rešitev je uporaba večih oznak različnih barv, ki so prilagojene za točno določeno okolje. Za izboljšanje robustnosti sledilnika bi bilo potrebno npr. vključiti dodatno metodo za sledenje, ki bi na drugi način sledila oznaki in tako zagotavljala dodatne informacije o poziciji oznake. Možne dodatne metode za integracijo v obstoječi sledilnik sta predvsem sledenje s pomočjo Kalmanovega filtra in optični tok, ki delujeta na osnovi gibanja. Z dodatkom sledenja gibanju dobi obstoječi sledilnik novo razsežnost in se lahko zanaša na barve ter gibanje skupaj, kar bi omogočilo večjo zanesljivost in izboljšalo robustnost. Potrebno bi bilo podrobneje raziskati metrike določanja zanesljivosti sledilnika, da bi lahko bolje opredelili zanesljivost sledenja. Poleg tega je potrebno narediti še veliko testiranj sistema, preden bi zadevo vgradili na pravi letalnik in ga preizkusili v realnih pogojih. Sistem bi morali testirati najprej na realnih posnetkih, ki so snemani iz kamere letalnika. Pri delovanju sistema v težjih pogojih bi bilo mogoče uporabiti še toplotno kamero, s katero bi bilo mogoče enostavno locirati oznako z dodanim toplotnim virom. Prostora za izboljšave je še veliko, ampak kljub temu je implementirani sistem pokazal dobre rezultate in dosegel cilj, ki smo si ga zastavili.

Literatura

[1] C-astral, bramor C4EYE, [Dostop: 10. december 2017].
URL `http://www.c-astral.com/en/products/bramor-c4eye`

[2] S. Huh, D. H. Shim, A vision-based automatic landing method for fixed-wing UAVs, Journal of Intelligent & Robotic Systems 57 (1) (2010) 217–231.

[3] O. Bourquardez, F. Chaumette, Visual servoing of an airplane for auto-landing, in: Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on, IEEE, 2007, pp. 1314–1319.

[4] W. Sai-fei, W. Xin-hua, B. Jun-jie, T. Qing-yan, Autonomous landing of a fixed wing uav with a ground-based visual guidance system, DEStech Transactions on Engineering and Technology Research (ICMITE2016).

[5] M. Laiacker, K. Kondak, M. Schwarzbach, T. Muskardin, Vision aided automatic landing system for fixed wing uav, in: Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on, IEEE, 2013, pp. 2971–2976.

[6] H. Uchiyama, E. Marchand, Object detection and pose tracking for augmented reality: Recent approaches, in: 18th Korea-Japan Joint Workshop on Frontiers of Computer Vision (FCV), 2012.

[7] C. Harris, M. Stephens, A combined corner and edge detector., in: Alvey vision conference, Vol. 15, Manchester, UK, 1988, pp. 10–5244.

[8] E. Rosten, T. Drummond, Machine learning for high-speed corner detection, Computer Vision–ECCV 2006 (2006) 430–443.

[9] D. G. Lowe, Distinctive image features from scale-invariant keypoints, International journal of computer vision 60 (2) (2004) 91–110.

[10] H. Bay, A. Ess, T. Tuytelaars, L. Van Gool, Speeded-up robust features (surf), Computer vision and image understanding 110 (3) (2008) 346–359.

[11] M. Agrawal, K. Konolige, M. R. Blas, Censure: Center surround extremas for realtime feature detection and matching, in: European Conference on Computer Vision, Springer, 2008, pp. 102–115.

[12] M. Li, C. Wei, Y. Yuan, Z. Cai, A survey of video object tracking, International Journal of Control and Automation 8 (9) (2015) 303–312.

[13] R. E. Kalman, et al., A new approach to linear filtering and prediction problems, Journal of basic Engineering 82 (1) (1960) 35–45.

[14] C. P. Robert, Monte carlo methods, Wiley Online Library, 2004.

[15] M. Fiala, Designing highly reliable fiducial markers, IEEE Transactions on Pattern analysis and machine intelligence 32 (7) (2010) 1317–1324.

[16] L. Calvet, P. Gurdjos, C. Griwodz, S. Gasparini, Detection and accurate localization of circular fiducials under highly challenging conditions, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 562–570.

[17] D. Wagner, T. Langlotz, D. Schmalstieg, Robust and unobtrusive marker tracking on mobile phones, in: Proceedings of the 7th IEEE/ACM International Symposium on Mixed and Augmented Reality, IEEE Computer Society, 2008, pp. 121–124.

[18] F. Bergamasco, A. Albarelli, E. Rodolà, A. Torsello, Rune-tag: A high accuracy fiducial marker with strong occlusion resilience, in: CVPR, 2011.

[19] L. B. Gatrell, W. A. Hoff, C. W. Sklair, Robust image features: Concentric contrasting circles and their image extraction, in: Robotics-DL tentative, International Society for Optics and Photonics, 1992, pp. 235–244.

[20] J. Köhler, A. Pagani, D. Stricker, Detection and identification techniques for markers used in computer vision, in: OASIcs-OpenAccess Series in Informatics, Vol. 19, Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2011.

[21] Love affair with cbir part 3, [Dostop: 10. december 2017].
URL `https://sbrakl.wordpress.com/2015/01/30/love-affair-with-cbir-part-3/`

[22] M. G. Nayagam, K. Ramar, A survey on real time object detection and tracking algorithms, Int. J. Appl. Eng. Res 10 (9) (2015) 8290–8297.

[23] Feature detection, [Dostop: 10. december 2017].
URL `https://goo.gl/FCEPTe`

[24] J. Canny, A computational approach to edge detection, IEEE Transactions on pattern analysis and machine intelligence (6) (1986) 679–698.

[25] J. Shi, et al., Good features to track, in: Computer Vision and Pattern Recognition, 1994. Proceedings CVPR'94., 1994 IEEE Computer Society Conference on, IEEE, 1994, pp. 593–600.

[26] M. Donoser, H. Bischof, Efficient maximally stable extremal region (MSER) tracking, in: Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on, Vol. 1, IEEE, 2006, pp. 553–560.

[27] S. Gauglitz, T. Höllerer, M. Turk, Evaluation of interest point detectors and feature descriptors for visual tracking, International journal of computer vision 94 (3) (2011) 335–360.

[28] M. Hirzer, Marker detection for augmented reality applications, in: Seminar/Project Image Analysis Graz, 2008, pp. 1–2.

[29] S. Siltanen, Theory and applications of marker-based augmented reality, 2012.

[30] C. Tomasi, T. Kanade, Detection and tracking of point features.

[31] A. Yilmaz, O. Javed, M. Shah, Object tracking: A survey, ACM computing surveys (CSUR) 38 (4) (2006) 13.

[32] S. C. Sen-Ching, C. Kamath, Robust techniques for background subtraction in urban traffic video, in: Electronic Imaging 2004, International Society for Optics and Photonics, 2004, pp. 881–892.

[33] Y. Cheng, Mean shift, mode seeking, and clustering, IEEE transactions on pattern analysis and machine intelligence 17 (8) (1995) 790–799.

[34] C. Rother, V. Kolmogorov, A. Blake, Grabcut: Interactive foreground extraction using iterated graph cuts, in: ACM transactions on graphics (TOG), Vol. 23, ACM, 2004, pp. 309–314.

[35] T. F. Chan, L. A. Vese, Active contours without edges, IEEE Transactions on image processing 10 (2) (2001) 266–277.

[36] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, Nature 521 (7553) (2015) 436–444.

[37] A. Krizhevsky, I. Sutskever, G. E. Hinton, Imagenet classification with deep convolutional neural networks, in: Advances in neural information processing systems, 2012, pp. 1097–1105.

[38] R. Girshick, J. Donahue, T. Darrell, J. Malik, Rich feature hierarchies for accurate object detection and semantic segmentation, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2014, pp. 580–587.

[39] S. Challa, Fundamentals of object tracking, Cambridge University Press, 2011.

[40] R. K. Rout, A survey on object detection and tracking algorithms, Ph.D. thesis (2013).

[41] Q. Yu, T. B. Dinh, G. Medioni, Online tracking and reacquisition using co-trained generative and discriminative trackers, in: European conference on computer vision, Springer, 2008, pp. 678–691.

[42] I. K. Sethi, R. Jain, Finding trajectories of feature points in a monocular image sequence, IEEE Transactions on pattern analysis and machine intelligence (1) (1987) 56–73.

[43] D. Comaniciu, V. Ramesh, P. Meer, Kernel-based object tracking, IEEE Transactions on pattern analysis and machine intelligence 25 (5) (2003) 564–577.

[44] S.-H. Cha, Comprehensive survey on distance/similarity measures between probability density functions, City 1 (2) (2007) 1.

[45] C. Cassisi, P. Montalto, M. Aliotta, A. Cannata, A. Pulvirenti, Similarity measures and dimensionality reduction techniques for time series data mining, in: Advances in data mining knowledge discovery and applications, InTech, 2012.

[46] T. F. Cootes, G. V. Wheeler, K. N. Walker, C. J. Taylor, View-based active appearance models, Image and vision computing 20 (9) (2002) 657–664.

[47] H. Nam, B. Han, Learning multi-domain convolutional neural networks for visual tracking, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 4293–4302.

[48] G. Zhu, F. Porikli, H. Li, Tracking randomly moving objects on edge box proposals, arXiv preprint arXiv:1507.08085.

[49] A. Lukežič, L. Č. Zajc, M. Kristan, Deformable parts correlation filters for robust visual tracking, IEEE Transactions on Cybernetics.

[50] Canny edge detection in C#, [Dostop: 10. december 2017].
URL `https://www.codeproject.com/Articles/93642/Canny-Edge-Detection-in-C`

[51] S. Suzuki, et al., Topological structural analysis of digitized binary images by border following, Computer vision, graphics, and image processing 30 (1) (1985) 32–46.

[52] A. W. Fitzgibbon, R. B. Fisher, et al., A buyer's guide to conic fitting, DAI Research paper.

[53] M. J. Swain, D. H. Ballard, Color indexing, International journal of computer vision 7 (1) (1991) 11–32.

[54] D. M. de Brito, V. Maracaja-Coutinho, S. T. de Farias, L. V. Batista, T. G. do Rêgo, A novel method to predict genomic islands based on mean shift clustering algorithm, PloS one 11 (1) (2016) e0146352.

[55] G. R. Bradski, Real time face and object tracking as a component of a perceptual user interface, in: Applications of Computer Vision, 1998. WACV'98. Proceedings., Fourth IEEE Workshop on, IEEE, 1998, pp. 214–219.

[56] G. Bishop, G. Welch, An introduction to the kalman filter, Proc of SIGGRAPH, Course 8 (27599-23175) (2001) 41.

[57] Aibu - image sequence annotator, [Dostop: 8. december 2017].
URL `https://github.com/votchallenge/aibu`

[58] Videezy, [Dostop: 10. december 2017].
URL `https://www.videezy.com/`

[59] Pexels videos, [Dostop: 10. december 2017].
URL `https://videos.pexels.com/`